

CA-IDMS®

Total Transparency User Guide 15.0



Computer Associates™

This documentation and related computer software program (hereinafter referred to as the "Documentation") is for the end user's informational purposes only and is subject to change or withdrawal by Computer Associates International, Inc. ("CA") at any time.

THIS DOCUMENTATION MAY NOT BE COPIED, TRANSFERRED, REPRODUCED, DISCLOSED, OR DUPLICATED, IN WHOLE OR IN PART, WITHOUT THE PRIOR WRITTEN CONSENT OF CA. THIS DOCUMENTATION IS PROPRIETARY INFORMATION OF CA AND PROTECTED BY THE COPYRIGHT LAWS OF THE UNITED STATES AND INTERNATIONAL TREATIES.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION "AS IS" WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO THE END USER OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED OF SUCH LOSS OR DAMAGE.

THE USE OF ANY PRODUCT REFERENCED IN THIS DOCUMENTATION AND THIS DOCUMENTATION IS GOVERNED BY THE END USER'S APPLICABLE LICENSE AGREEMENT.

The manufacturer of this documentation is Computer Associates International, Inc.

Provided with "Restricted Rights" as set forth in 48 C.F.R. Section 12.212, 48 C.F.R. Sections 52.227-19(c)(1) and (2) or DFARS Section 252.227.7013(c)(1)(ii) or applicable successor provisions.

First Edition, December 2000

© 2000 Computer Associates International, Inc.
One Computer Associates Plaza, Islandia, NY 11749
All rights reserved.

All trademarks, trade names, service marks, or logos referenced herein belong to their respective companies.

Contents

How to use this Document	vii
Chapter 1. Overview of the CA-IDMS/Total Transparency	1-1
1.1 What is the CA-IDMS/Total Transparency?	1-3
1.1.1 CA-IDMS/Total Transparency as a conversion tool	1-3
1.1.2 CA-IDMS/Total Transparency as a runtime tool	1-3
1.2 Operating environments supported	1-4
1.3 Components of the CA-IDMS/Total Transparency	1-5
1.3.1 The generator: ETOTMAIN	1-5
1.3.2 The loader: ETOTLOAD	1-7
1.3.3 The runtime control table: ETOTTBL	1-7
1.3.4 The runtime interface	1-7
1.4 Implementing the transparency	1-9
1.4.1 Data description phase	1-9
1.4.2 Database conversion phase	1-9
1.4.3 Runtime operations phase	1-11
1.5 Conversion of Total data structures to CA-IDMS/DB	1-13
1.5.1 Total files relate to CA-IDMS/DB record types	1-13
1.5.2 Total data items relate to CA-IDMS/DB elements	1-13
1.5.3 Total master file control keys relate to CA-IDMS/DB CALC keys	1-13
1.5.4 Total linkpath fields relate to CA-IDMS/DB pointers	1-13
1.5.5 Total linkpaths relate to CA-IDMS/DB sets	1-13
1.5.6 Total relationships	1-14
1.5.7 CA-IDMS/DB relationships	1-14
1.6 Support for Total	1-15
1.6.1 Total releases supported	1-15
1.6.2 Total features supported	1-15
1.6.3 Total features not supported	1-16
Chapter 2. Data Description Phase	2-1
2.1 Steps of the data description phase	2-3
2.1.1 Step 1 — Select a Total database	2-3
2.1.2 Step 2 — Prepare Total DDL statements	2-3
2.1.3 Step 3 — Prepare the physical CA-IDMS/DB database	2-4
2.1.4 Step 4 — Prepare the logical CA-IDMS/DB database	2-4
2.2 Generator control statements	2-5
2.2.1 Summary of statements	2-5
2.2.2 Limits on control statements	2-6
2.2.3 SYNONYMS statement	2-6
2.2.4 AREA statement	2-8
2.2.5 SCHEMA statement	2-9
2.2.6 SUBSCHEMA statement	2-11
2.2.7 USAGE-MODE statement	2-12
2.2.8 LOAD statement	2-15
Chapter 3. Database Conversion Phase	3-1
3.1 Steps of the database conversion phase	3-3

3.2 Step 1: Transparency generator (ETOTMAIN)	3-4
3.3 Step 2: Schema preparation	3-13
3.4 Step 3: Physical database preparation	3-14
3.5 Step 4: Subschema preparation	3-16
3.6 Step 5: CA-IDMS/DB database initialization	3-18
3.7 Step 6: Total database unload utility	3-19
3.8 Step 7: Transparency loader (ETOTLOAD)	3-20
Chapter 4. Runtime Operations Phase	4-1
4.1 Runtime steps	4-3
Appendix A. User Exits	A-1
A.1 DATBASBF and DATBASAF exits	A-3
A.2 Sample batch exit module	A-4
A.3 Sample CICS exit module	A-8
Appendix B. Generator Messages	B-1
B.1 About generator messages	B-3
B.2 Error messages	B-4
B.3 Warning message	B-7
B.4 Fatal messages	B-8
Appendix C. Loader Messages	C-1
C.1 Loader messages issued	C-3
Appendix D. Runtime Messages	D-1
D.1 Three classes of error codes	D-3
D.2 Total DML syntax error codes	D-4
D.3 Error conditions detected by CA-IDMS/DB	D-5
D.4 Transparency errors	D-8
Appendix E. Increasing Generator Input Limits	E-1
E.1 ETOTMAIN input limits	E-3
E.1.1 Limits on Total DDL statements	E-3
E.1.2 Limits on transparency generator control statements	E-3
Appendix F. OS/390 Job Control Language	F-1
F.1 Generate (ETOTMAIN)	F-3
F.2 Assembling and link editing ETOTTBL	F-5
F.3 Compiling and link editing ETOTLOAD	F-6
F.4 Loading the database with ETOTLOAD	F-8
F.5 Assembling the CICS transparency interface (ETOTCINT)	F-10
F.6 Creating the CICS CA-IDMS/DB Interface (IDMSINTC)	F-11
F.7 Batch application program link edit	F-12
F.8 CICS application program link edit	F-13
F.9 Runtime JCL	F-14
Appendix G. VSE/ESA Job Control Language	G-1
G.1 Generate (ETOTMAIN)	G-2
G.2 Assembling and link editing ETOTTBL	G-4
G.3 Compiling and link editing ETOTLOAD	G-5

G.4 Loading the database with ETOTLOAD	G-6
G.5 Assembling the CICS/transparency interface (ETOTCINT)	G-8
G.6 Creating the CICS CA-IDMS/DB interface (IDMSINTC)	G-9
G.7 Batch application program link edit	G-10
G.8 CICS application program link edit	G-11
G.9 Runtime JCL	G-12
Index	X-1

How to use this Document

What this document is about

This document provides information on converting a Total database to a CA-IDMS/DB database. Total is a product of Cincom Systems, Incorporated. CA-IDMS/Total Transparency is the tool you use for database conversion and runtime operations.

Who should use this document

This document is for database administrators converting Total databases to CA-IDMS/DB databases and for application developers running Total application programs against the converted databases.

You should be familiar with CA-IDMS/DB concepts and facilities before you use this document and the CA-IDMS/Total Transparency product.

What this document contains

This document contains four chapters and seven appendixes:

- A System Overview (Chapter 1) — Explanation of the CA-IDMS/Total Transparency product and its phases, and discussion of how CA-IDMS/Total Transparency concepts relate to Total concepts
- Data Description Phase (Chapter 2) — Instructions for converting Total data definitions to CA-IDMS/DB data definitions
- Database Conversion Phase (Chapter 3) — Instructions for converting a Total database to a CA-IDMS/DB database
- Runtime Operations Phase (Chapter 4) — Instructions for running existing Total applications against a CA-IDMS/DB database
- User Exits (Appendix A) — Explanation of the two CA-IDMS/Total Transparency user exits; includes a batch program illustrating their use
- Generator Messages (Appendix B) — Listing of the messages issued by the generator of CA-IDMS/Total Transparency
- Loader Messages (Appendix C) — Listing of ETOTLOAD messages
- Runtime Messages (Appendix D) — Discussion of runtime error codes including a list of USxx codes and table showing corresponding status codes for Total and CA-IDMS/DB
- Increasing Generator Input Limits (Appendix E) — How to increase limits for large database conversions
- OS/390 Job Control Language (Appendix F) — OS/390 JCL templates for all CA-IDMS/Total Transparency jobs
- VSE/ESA Job Control Language (Appendix G) — VSE/ESA JCL templates for all CA-IDMS/Total Transparency jobs

Related documents

Before you use the CA-IDMS/Total Transparency product, you should read the following documents:

- *CA-IDMS Database Design Guide*

As you use the transparency, you should have the following CA-IDMS documents available for reference:

- *CA-IDMS Messages and Codes*
- *CA-IDMS Installation and Maintenance Guide - OS/390*
- *CA-IDMS Installation and Maintenance Guide - VSE/ESA*
- *CA-IDMS System Generation*
- *CA-IDMS System Operations*
- *CA-IDMS Utilities*
- *CA-IDMS Database Administration*
- *CA-IDMS IDD DDDL Reference Guide*
- *CA-IDMS Command Facility*

How to proceed

1. Read Chapter 1, “Overview of the CA-IDMS/Total Transparency” on page 1-1.
2. Choose the database you will work with. For your first use of the CA-IDMS/Total Transparency product, select a small database or a subset of a larger database. This pilot project will give you useful experience for dealing with large databases.
3. Follow the directions in Chapters 2 through 4.

Chapter 1. Overview of the CA-IDMS/Total Transparency

1.1 What is the CA-IDMS/Total Transparency?	1-3
1.1.1 CA-IDMS/Total Transparency as a conversion tool	1-3
1.1.2 CA-IDMS/Total Transparency as a runtime tool	1-3
1.2 Operating environments supported	1-4
1.3 Components of the CA-IDMS/Total Transparency	1-5
1.3.1 The generator: ETOTMAIN	1-5
1.3.2 The loader: ETOTLOAD	1-7
1.3.3 The runtime control table: ETOTTBL	1-7
1.3.4 The runtime interface	1-7
1.4 Implementing the transparency	1-9
1.4.1 Data description phase	1-9
1.4.2 Database conversion phase	1-9
1.4.3 Runtime operations phase	1-11
1.5 Conversion of Total data structures to CA-IDMS/DB	1-13
1.5.1 Total files relate to CA-IDMS/DB record types	1-13
1.5.2 Total data items relate to CA-IDMS/DB elements	1-13
1.5.3 Total master file control keys relate to CA-IDMS/DB CALC keys	1-13
1.5.4 Total linkpath fields relate to CA-IDMS/DB pointers	1-13
1.5.5 Total linkpaths relate to CA-IDMS/DB sets	1-13
1.5.6 Total relationships	1-14
1.5.7 CA-IDMS/DB relationships	1-14
1.6 Support for Total	1-15
1.6.1 Total releases supported	1-15
1.6.2 Total features supported	1-15
1.6.3 Total features not supported	1-16

1.1 What is the CA-IDMS/Total Transparency?

The CA-IDMS/Total Transparency is a product that helps users of the Total Database Management System convert to a CA-IDMS/DB database environment. Through the transparency, Total users can:

- Convert an existing Total database to a CA-IDMS/DB database
- Run existing Total application programs against the new CA-IDMS/DB database

1.1.1 CA-IDMS/Total Transparency as a conversion tool

To convert a Total database to a CA-IDMS/DB database, the transparency does the following:

- Translates Total data definition language (DDL) into the logical DDL needed for a CA-IDMS/DB database
- Loads the unloaded Total database to a CA-IDMS/DB database

1.1.2 CA-IDMS/Total Transparency as a runtime tool

To enable Total application programs to run against the CA-IDMS/DB database, the transparency provides a runtime interface. This interface *replaces* all Total software and performs the following functions:

- Simulates the Total database processing environment to satisfy database requests issued by Total applications
- Converts Total data manipulation language (DML) calls into CA-IDMS DML calls
- Represents CA-IDMS/DB data structures as Total data structures
- Accepts Total database requests from both batch and CICS programs. CICS programs can continue to issue requests at both the command level and the macro level.

Note: The exception is CICS 3.22, which supports only command level requests.

Note: If you are a MANTIS user using CICS 3.22, you must rename the ETOTMINT load module to ETOTINT2, and ETOTINT1 to ETOTMINT.

1.2 Operating environments supported

The transparency runs in the IBM OS/390 and VSE/ESA operating environments, both batch and CICS users.

1.3 Components of the CA-IDMS/Total Transparency

The major components of the CA-IDMS/Total Transparency are as follows:

- Generator
- Load module
- Runtime control table
- Runtime interface

1.3.1 The generator: ETOTMAIN

The generator (ETOTMAIN) is a transparency program that does the following:

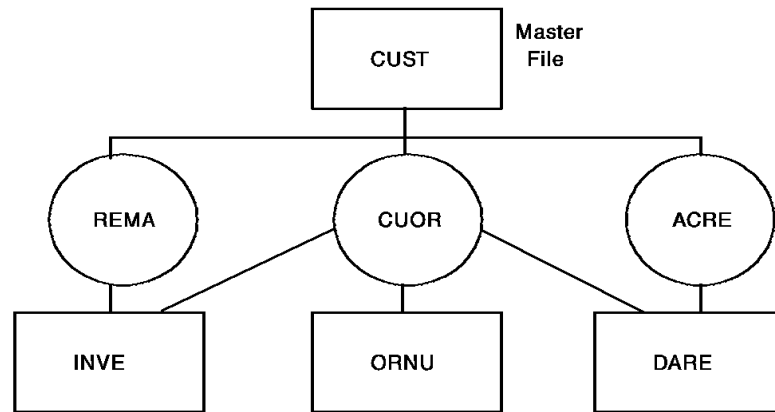
- Translates Total DDL into CA-IDMS/DB DDL (schema and subschema descriptions)
- Generates a customized load module (ETOTLOAD)
- Generates a customized table (ETOTTBL) that defines the database to the runtime interface

The following table describes input to the generator.

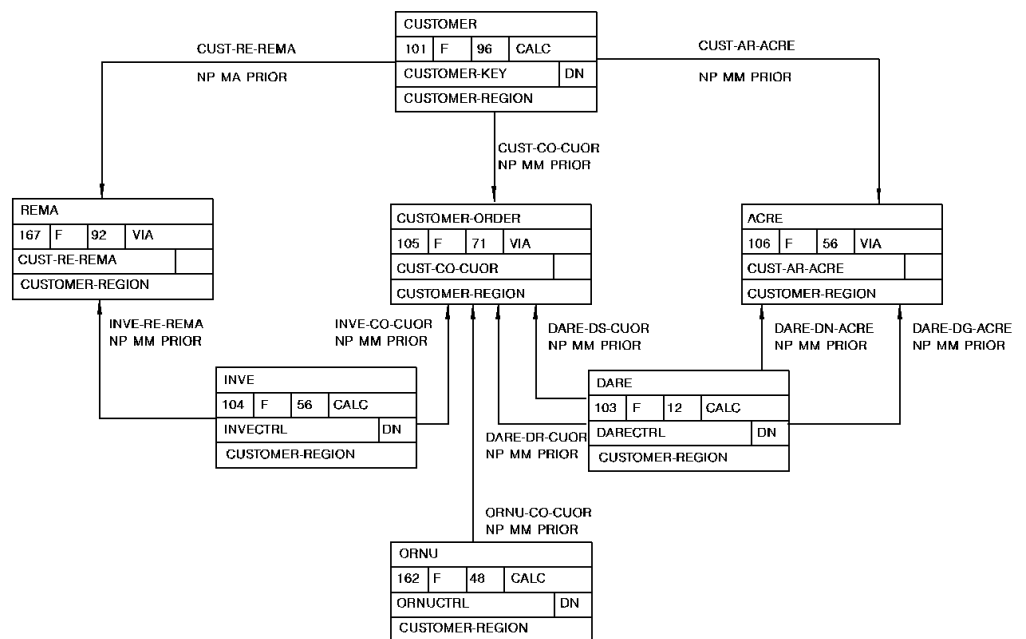
Input	Description
Total DDL	Defines the entire Total database being converted
Control statements that you prepare	Supply information about the CA-IDMS/DB database being created
Loader (ETOTLOAD) skeleton containing source COBOL statements	Reads unloaded Total data and loads the CA-IDMS/DB database
Runtime control table (ETOTTBL) skeleton	Creates source Assembler statements that provide interface information about the Total database referenced by Total application programs

Equivalent Total and CA-IDMS/DB data structures: The following diagrams are of a sample Total database and the corresponding CA-IDMS/DB database. Note that the CUSTOMER and CUSTOMER-ORDER CA-IDMS/DB records correspond to the Total master file CUST and the CUOR variable file. Synonyms are provided to take advantage of the longer names permitted under CA-IDMS/DB.

TOTAL database structure



CA-IDMS/DB database structure



1.3.2 The loader: ETOTLOAD

The loader (ETOTLOAD) is a customized COBOL source module output by the generator. The loader contains code to read data unloaded from the Total database and to load the corresponding CA-IDMS/DB database.

1.3.3 The runtime control table: ETOTTBL

The runtime control table (ETOTTBL) is a customized assembler source module output by the generator. Using Total DDL and user-supplied control statements, the generator copies arrays into the WORKING-STORAGE SECTION of ETOTTBL. These arrays describe Total files and their equivalent CA-IDMS/DB record types and set relationships.

At runtime, ETOTTBL replaces the Total DBMOD and provides the runtime interface with the control information necessary to:

- Construct the Total records and relationships expected by the Total application program
- Reconstruct CA-IDMS/DB records to be stored in the CA-IDMS/DB database

1.3.4 The runtime interface

The runtime interface translates Total database calls and passes the translated calls to CA-IDMS/DB.

Operation	How it works
Translate Total calls	The object module ETOTTRAN translates calls. ETOTTRAN can be link edited to the Total application. If it is not link edited, it will be loaded at runtime. You might want to link ETOTTRAN with batch programs, but dynamically load ETOTTRAN when you use online CICS applications.

Operation	How it works
Pass translated calls to CA-IDMS/DB	<p>Total batch applications</p> <p>Total batch applications use the object module ETOTBINT. ETOTBINT is link edited with the Total application and CA-IDMS/DB to access the CA-IDMS/DB database. The runtime control table is loaded by ETOTBINT at runtime.</p> <p>Total CICS applications</p> <p>Total CICS applications use the macro ETOTCINT to pass calls to program IDMSINTC. IDMSINTC, in turn, passes the calls to CA-IDMS/DB. ETOTCINT must be assembled before runtime. The assembled ETOTCINT is link edited with the Total application. ETOTTRAN is link edited as part of the IDMSINTC load module.</p>

The runtime interface uses the generated subschema to access the CA-IDMS/DB database. This subschema provides access to all elements, record types, set types, and areas defined in the generated schema.

The Mixed Page Group Binds Allowed feature may not be used with CA-IDMS/Total Transparency.

1.4 Implementing the transparency

The transparency has three phases:

1. Data description
2. Database conversion
3. Runtime operations

1.4.1 Data description phase

In the data description phase, you do the following:

1. Select a Total database for conversion
2. Check that you have the appropriate Total DDL to describe the database selected
3. Prepare control statements; these control statements describe the *logical* CA-IDMS/DB environment that results from database conversion
4. Prepare CA-IDMS DDL statements that describe the *physical* CA-IDMS/DB environment

You input the Total DDL and control statements in the the database conversion phase.

►► For detailed information about the data description phase, see Chapter 2, “Data Description Phase” on page 2-1.

1.4.2 Database conversion phase

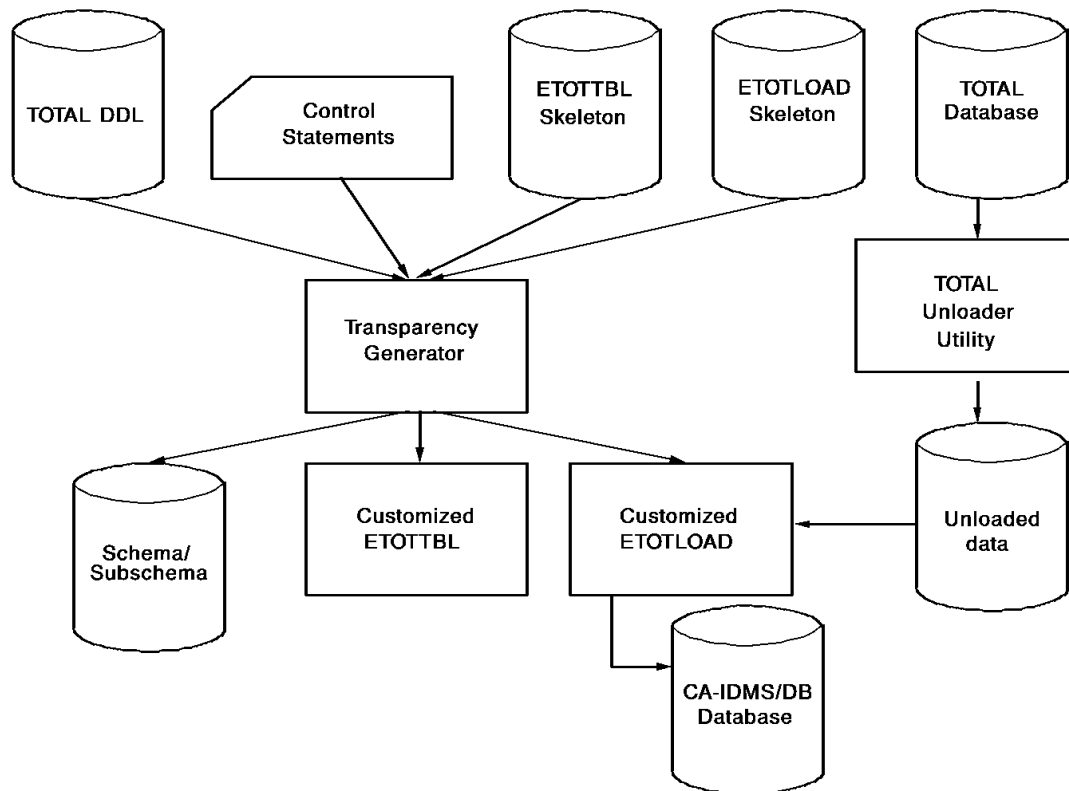
Database conversion programs: In the database conversion phase, you convert your Total database to a CA-IDMS/DB database using the programs listed in the following table.

Program	Input	Ouput
Generator (ETOTMAIN)	■ User-prepared control statements	■ Source DDL for the schema and subschema
	■ Total DDL	■ Customized load module
	■ Skeletons for the loader (ETOTLOAD)	■ Customized runtime control table module
	■ Runtime control table (ETOTTBL) modules	

Program	Input	Ouput
CA-IDMS/DB schema compiler (IDMSCHEM)	Schema DDL produced by the transparency generator	Compiled schema descriptions entered in the dictionary (IDD)
CA-IDMS Command Facility (IDMSBCF)	Physical DDL statements that define segments, a database name table, and the DMCL ►► For physical DDL syntax, refer to <i>CA-IDMS Database Administration</i> . For information about using IDMSBCF to submit physical DDL statements, refer to <i>CA-IDMS Command Facility</i> .	<ul style="list-style-type: none"> ■ Compiled DMCL descriptions entered in the catalog (DDL CAT) area of the system dictionary ■ DMCL load module ■ Database name table load module
CA-IDMS/DB subschema compiler (IDMSUBSC)	Subschema DDL produced by the generator	<ul style="list-style-type: none"> ■ Compiled subschema descriptions entered in the dictionary (IDD) ■ Subschema load module
CA-IDMS Command Facility (IDMSBCF)	FORMAT utility statement; use the names of the files included in the DMCL load module ►► For FORMAT syntax, refer to <i>CA-IDMS Utilities</i> . For information on using IDMSBCF to submit FORMAT, refer to <i>CA-IDMS Command Facility</i> .	Formatted CA-IDMS/DB database
Total unload utility (CSITULOD)	Parameters and data in the Total database	Unloaded data
Customized loader (ETOTLOAD)	Unloaded data	Data in the CA-IDMS/DB database

When you finish the conversion phase, the CA-IDMS/DB database is available for processing. You are ready for the final transparency phase, runtime operations.

Illustration of the database conversion phase: The following figure illustrates the database conversion phase.



►► For detailed information about the database conversion phase, see Chapter 3, “Database Conversion Phase” on page 3-1.

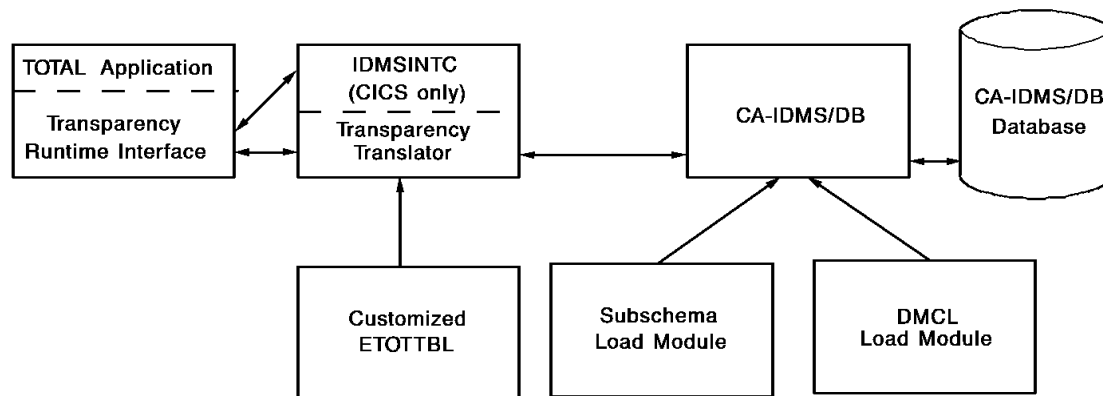
1.4.3 Runtime operations phase

In the runtime operations phase, you relink your Total and CICS applications to run against the new CA-IDMS/DB database.

Runtime operations include the following:

- Developing and running native CA-IDMS/DB application programs (refer to the appropriate CA-IDMS/DB documents)
- Running Total and CICS application programs by using the runtime interface

Illustration of the runtime operations phase: The following figure illustrates the runtime operations phase.



►► For detailed information about the runtime operations phase, see Chapter 4, “Runtime Operations Phase” on page 4-1.

1.5 Conversion of Total data structures to CA-IDMS/DB

To convert a Total database to CA-IDMS/DB, the transparency converts Total data structures into CA-IDMS/DB data structures. This section provides information about the CA-IDMS/DB equivalents of Total data structures.

1.5.1 Total files relate to CA-IDMS/DB record types

A CA-IDMS/DB record type represents a collection of similar record occurrences. Record types are defined to CA-IDMS/DB by schema record descriptions.

1.5.2 Total data items relate to CA-IDMS/DB elements

CA-IDMS/DB elements are the individual units of user data that make up a record. Elements and their characteristics are defined in schema record descriptions.

1.5.3 Total master file control keys relate to CA-IDMS/DB CALC keys

A CA-IDMS/DB CALC key is an element designated as the value within a record used to determine the location of the record in a database area.

The transparency assigns a location mode of CALC to all records derived from Total master files. The transparency assigns a location mode of VIA to all records derived from Total variable files, with the exception of records in coded variable files for which the primary linkpath is not included in the base portion of the file. Records included in these coded variable files are assigned a location mode of DIRECT.

1.5.4 Total linkpath fields relate to CA-IDMS/DB pointers

A CA-IDMS/DB pointer links the owner and member records of a CA-IDMS/DB set. Note that CA-IDMS/DB pointers, unlike Total linkpath fields, are maintained in a record's prefix rather than in the record itself. The transparency uses Total linkpath field information to establish the necessary pointers in the record prefix for each record. The transparency does not retain Total linkpath fields in inverted records.

1.5.5 Total linkpaths relate to CA-IDMS/DB sets

Identifies existing relationships: When converting a Total database to a CA-IDMS/DB database, the transparency uses Total linkpath information to identify each relationship that exists among given Total files. For each such relationship found in Total DDL, the transparency defines a set in the generator-produced CA-IDMS/DB schema and subschema.

Converting to a set: When converting a linkpath to a set, the transparency does the following:

- Designates the Total master file as the owner record type

- Designates the associated Total variable file as the member record type
- Establishes appropriate pointers from the linkpath fields used to relate a master file to a variable file and to chain records in a variable file

PRIOR set order is assigned: The transparency assigns a set order of PRIOR to each CA-IDMS/DB set established. It assigns a set membership option of MANDATORY AUTOMATIC for the primary linkpath and a set membership option of MANDATORY MANUAL for all other CA-IDMS/DB member record types that it describes in the schema.

►► For a discussion of CA-IDMS/DB set order and set membership options, refer to *CA-IDMS Database Administration*.

Records in a Total variable file are stored via the CA-IDMS/DB set established to correspond to the primary linkpath for that file.

1.5.6 Total relationships

A Total file definition includes not only descriptions of user data but also descriptions of any one-to-many relationships that exist with other files. This is reflected in the presence of linkpath fields and the designation of files as either master files or variable files.

Linkpath fields are defined: To associate a master file with a variable file, linkpath fields are defined. A linkpath field in a master record points to the first and last of a chain of related records in a variable file. A linkpath field in a variable record points to the next and prior variable record in the chain. Total maintains the one-to-many relationship as part of the data in the record.

1.5.7 CA-IDMS/DB relationships

While the relationships among CA-IDMS/DB record types are similar in concept to those among Total files, these relationships are defined and maintained differently. Under CA-IDMS/DB, a one-to-many relationship between two or more record types is expressed by a set structure. A set structure is defined and maintained apart from the record definitions.

Set is defined for each relationship For each relationship between two or more CA-IDMS/DB record types, a set is defined. One record type is designated as the owner of the set and the other record type(s) is designated as the member. A given set occurrence consists of one occurrence of the owner record type and any number of occurrences of the member record type(s).:

Owner and members are linked by pointers The owner and members of a set are linked by the pointers maintained in the record prefixes. The record prefix always contains a pointer linking the record to the next record in the set, and can contain pointers linking the record to the prior, last, and/or owner record in the set.

1.6 Support for Total

This section provides information on Total releases supported and Total features supported/not supported under the CA-IDMS/Total Transparency.

1.6.1 Total releases supported

Release number	Support
8	Direct support
5,6, and 7	Support through user exits ►► For information on user exits, see Appendix A, “User Exits” on page A-1.

1.6.2 Total features supported

The transparency supports most Total DML statements that access and update the database. Almost all Total DML statements supported by the transparency translate directly to corresponding CA-IDMS DML statements. The Total DML statements described in the following table are translated to CA-IDMS, but with some variations in function. The variations are explained in the table.

Note: Total DML recovery and logging functions are not converted to CA-IDMS/DB. CA-IDMS/DB provides recovery utilities and journaling options that adequately replace these Total DML functions.

For this Total DML statement	CA-IDMS/Total Transparency does the following:
CNTRL (purge)	Ignores the command
COMIT	Issues a COMMIT ALL
ENDLG	Issues a COMMIT
FREEX	Issues a COMMIT ALL
QMARK	Issues a COMMIT
QUIET	Issues a COMMIT

For this Total DML statement	CA-IDMS/Total Transparency does the following:
SINON and OPENX	<ul style="list-style-type: none">■ Issues A BIND RUN UNIT statement■ Issues BIND statements for all CA-IDMS/DB record types described in the subschema■ Issues a READY statement for all CA-IDMS/DB database areas included in the subschema <p>After the runtime system processes a Total SINON or OPENX command, it will not reissue the CA-IDMS/DB BIND/READY sequence.</p> <p>When a SINON command is issued, the runtime interface performs the following actions:</p> <ul style="list-style-type: none">■ Saves the name of the requesting task for use with a subsequent OPENX command■ Loads the runtime control table (ETOTTBL) using the table name referenced by the DBMOD parameter of the SINON statement
CLOX and SINOF	<p>Issues a FINISH statement</p> <p>FINISH releases all database areas (and therefore all database records) from program control. If a SINOF command is preceded by a CLOX command, the runtime interface does not issue a FINISH in response to the SINOF command.</p>
RQLOC (batch)	<p>Accepts a key value from the Total application and returns a suitable DBKEY value. Note that for RQLOC, the module IDMSCALC is required. If IDMSCALC is not link edited to ETOTBINT, it is loaded the first time the RQLOC function is invoked.</p>

1.6.3 Total features not supported

The transparency's runtime interface does not support Total DML functions that are inappropriate in a CA-IDMS/DB database environment. Information about unsupported functions is provided in the following table.

Unsupported Total DML statement or function	More information
ENDTO MARK1 RQLOC (CICS) RSTAT WRITD	<p>These statements are related to recovery and logging. The runtime system returns an error for all of these statements. You can alter programs containing these statements in any of these three ways:</p> <ul style="list-style-type: none"> ■ Use a user exit to perform the function ■ Check for the error status (FUNC) ■ Remove the statement <p>Additionally, the logging options parameter of the SINON command and the **REST** parameter in Total DML commands are supported for retrieval only. When updating the database, these parameters are ignored.</p> <p>The **REST** parameter <i>can</i> be used in Total retrieval DML to indicate that the entire physical record (as defined in the DBMOD) is to be used, rather than just an element list of the record.</p>
SKIP option of Total RDNXT and FINDX	SKIP option of Total RDNXT and FINDX (for skipping unused records)
Some OPENX options	<p>OPENX options that allow the user to ready Total files residing in the same area in different usage modes. The CA-IDMS/DB database is readied for processing at the area level and each area can contain multiple record types. All CA-IDMS/DB record types within a given area are accessed in the same usage mode. This approach differs from that of the Total Database Management System, which readies the database at the file level. Since each Total file contains records of only one type, each Total record type can be accessed in a different usage mode.</p>

Unsupported Total DML statement or function	More information
Some ADDV statements	<p data-bbox="743 373 1432 604">ADDV statements issued for a coded variable record for which a record code and coded linkpath field were not defined in Total DDL. To override this restriction, you must modify the Total DDL used to generate the runtime control table prior to submitting the Total DDL to the transparency generator (see 2.1.2, “Step 2 — Prepare Total DDL statements” on page 2-3).</p> <p data-bbox="743 625 1336 688">Do the following to define the coded variable file as a standard variable file:</p> <ol data-bbox="760 709 1404 913" style="list-style-type: none"><li data-bbox="760 709 1404 772">1. Delete the Total DDL statement that defines the overlay element for the coded file<li data-bbox="760 793 1404 913">2. Define as a group item the data item designated as the overlay element for the coded file; the group item replacing the overlay element should reference all data items following the deleted statement

Chapter 2. Data Description Phase

2.1 Steps of the data description phase	2-3
2.1.1 Step 1 — Select a Total database	2-3
2.1.2 Step 2 — Prepare Total DDL statements	2-3
2.1.3 Step 3 — Prepare the physical CA-IDMS/DB database	2-4
2.1.4 Step 4 — Prepare the logical CA-IDMS/DB database	2-4
2.2 Generator control statements	2-5
2.2.1 Summary of statements	2-5
2.2.2 Limits on control statements	2-6
2.2.3 SYNONYMS statement	2-6
2.2.4 AREA statement	2-8
2.2.5 SCHEMA statement	2-9
2.2.6 SUBSCHEMA statement	2-11
2.2.7 USAGE-MODE statement	2-12
2.2.8 LOAD statement	2-15

2.1 Steps of the data description phase

The data description phase of transparency implementation is a planning and preparatory phase. It involves the following four steps:

1. Selecting a Total database for conversion
2. Preparing Total DDL statements that describe the Total database
3. Preparing the statements that describe the *logical* CA-IDMS/DB database
4. Preparing the CA-IDMS DDL statements that describe the *physical* CA-IDMS/DB database

Generator input: Later, during the database conversion phase, the statements you've prepared, the skeleton loader module (ETOTLOAD), and the skeleton runtime control table (ETOTTBL) are input to the generator. The generator uses this information to produce CA-IDMS/DB schema and subschema descriptions and to produce both a customized loader and a customized runtime control table.

2.1.1 Step 1 — Select a Total database

To convert to a CA-IDMS/DB environment, you must convert several databases. For the first conversion, it is recommended that you choose a small database (or, if necessary, a subset of a larger one) as a pilot project. This will allow you to gain experience, knowledge, and confidence for the full conversion.

2.1.2 Step 2 — Prepare Total DDL statements

After you select the Total database you want to convert, make sure you have all of the relevant Total DDL in a file. This file is part of the input to the generator.

If you are converting an entire database, you may already have the input file you need. If you are working with part of a database, however, you must prepare the input file appropriate to your conversion.

Comments in the Total DDL: If column 1 of a Total DDL input statement is blank, the generator considers the statement to be a comment and ignores it. This allows you to retain all of the original source DDL while selectively disabling some of the statements.

Limits on the Total DDL: At installation, the generator is set up to handle Total DDL input for up to 300 files, 5900 elements, 500 linkpaths, and 500 record codes. These limits are usually adequate for initial use of the transparency. If your DDL file exceeds these limits, see Appendix E, "Increasing Generator Input Limits" on page E-1 for information on increasing the limits.

2.1.3 Step 3 — Prepare the physical CA-IDMS/DB database

In this step you prepare physical CA-IDMS/DB DDL statements. These statements describe the *physical* component of the CA-IDMS/DB database. The physical component includes:

- **Segments** — collections of files and areas
- **Database name table** — names of databases and segments
- **Device-media control language (DMCL)** — buffers, journals, identification of a database name table, identification of segments for inclusion in the database, file overrides, and area overrides

►► For information on the syntax for CA-IDMS/DB physical DDL statements, refer to *CA-IDMS Database Administration*. For information on the CA-IDMS Command Facility which you use to submit physical DDL statements, refer to *CA-IDMS Command Facility*.

2.1.4 Step 4 — Prepare the logical CA-IDMS/DB database

In this step you prepare generator control statements. These statements describe the *logical* component of the CA-IDMS/DB database (the schema and subschema).

►► Detailed information for each generator control statement is presented in the next section.

The control statements and the Total DDL statements you prepared in step 2 are used to:

- Produce a complete schema description that reflects the Total database
- Produce complete subschema descriptions that reflect the Total database
- Build the loader module (ETOTLOAD); ETOTLOAD contains the information necessary to establish correspondence between the structure of the unloaded Total database and that of the CA-IDMS/DB database
- Build a runtime control table (ETOTTBL); ETOTTBL replaces the Total DBMOD; ETOTTBL contains information that associates database requests in Total programs with the appropriate CA-IDMS/DB data descriptions and relationships

Note: After you code the control statements you can move on to the next transparency phase (see Chapter 3, “Database Conversion Phase” on page 3-1).

2.2 Generator control statements

This section provides a table summarizing the generator control statements and a detailed description of each statement, including syntax and parameter descriptions.

2.2.1 Summary of statements

The following table describes each generator control statement and specifies whether it is required or optional.

Statement	Type	Function
SYNONYMS	Optional	Assigns synonyms to Total element and file names. If this statement is not input, the generator assigns existing Total names to the corresponding CA-IDMS/DB elements and records in the schema. To simplify the pilot project, we recommend that you not use synonyms.
AREA	Optional	Assigns Total files to user-specified CA-IDMS/DB areas. If this statement is not included, the CA-IDMS/DB record types that correspond to Total files will all be contained in the first area named in the SCHEMA statement.
SCHEMA	Required	Provides the generator with the SCHEMA, FILE, and AREA components of the CA-IDMS/DB schema description.
SUBSCHEMA	Required	Provides the generator with the subschema description. The SUBSCHEMA statement allows you to name the subschema and to include informational entries in the subschema.
USAGE-MODE	Optional	Specifies CA-IDMS/DB ready options for database areas that the runtime interface accesses in a usage mode other than shared update.

Statement	Type	Function
LOAD	Optional	Specifies whether the unloaded Total data contains master file control keys and linkpath information, and specifies the logical record length of the unloaded Total data if it is other than 120 bytes. You need to include this statement only if the unloaded Total data will contain master file control keys and linkpath information, or if the logical record length of the unloaded Total data is other than 120 bytes.

2.2.2 Limits on control statements

Installation limits set on generator control statements are as follows:

- 2000 16-byte synonyms (files, records, sets)
- 1000 32-byte synonyms (elements)
- 2000 usage mode statements
- 500 schema/subschema statements

►► For information on increasing the limit on schema and subschema statements (up to 5000), see Appendix E, “Increasing Generator Input Limits” on page E-1.

2.2.3 SYNONYMS statement

The SYNONYMS statement allows you to take advantage of the longer element, record, and set names permitted under CA-IDMS/DB.

Assigns existing Total name: When the transparency generates schema and subschema descriptions, it assigns the existing Total name to the corresponding CA-IDMS/DB name. The Total file name is assigned to the CA-IDMS/DB record type. The Total data item name is assigned to the CA-IDMS/DB element.

Size limits: The following are the size limits that apply to Total and CA-IDMS/DB names:

- A Total file name allows 4 characters; the corresponding CA-IDMS/DB record type name allows 16 characters
- A Total data item name allows 8 characters; the corresponding CA-IDMS/DB element name allows 32 characters

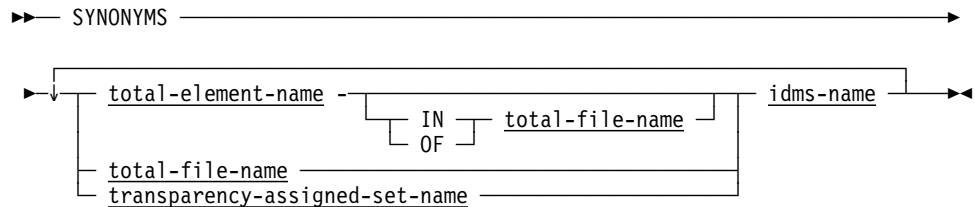
With the SYNONYMS statement, you can override the transparency's assignment of Total element and file names to CA-IDMS/DB elements and records.

You can also use the SYNONYMS statement to override the set names that the transparency assigns to CA-IDMS/DB sets established to represent relationships between Total master and variable files. The transparency derives CA-IDMS/DB set names from the applicable Total master file, linkpath, and variable file names.

Example of the transparency assigning a set name: The following example shows how a CA-IDMS/DB set name is derived from Total master file, linkpath, and variable file names.

Master file is CUST	CUST
+	
Linkpath is CO	CO
+	
Variable file is CUOR	+ CUOR
<hr/>	
CA-IDMS/DB set name	= CUST-CO-CUOR

Syntax



Parameters

SYNONYMS

Required keyword; must occupy a line by itself and begin in column 1.

total-element-name/total-file-name/transparency-assigned-set-name

Specifies the Total element or file name or the transparency-derived set name for which a synonym is to be used; this clause must begin in column 8.

total-element-name

8-character element name specified in the input Total DDL.

total-file-name

4-character file name specified in the input Total DDL.

transparency-assigned-set-name

Set name derived by CA-IDMS/Total Transparency for a set established to correspond to a Total master file/variable file relationship.

IN/OF total-file-name

Identifies the Total file for which *total-element-name* is defined in the Total DDL. This parameter must be specified if *total-element-name* is not unique to a given Total file (that is, if *total-element-name* exists for more than one file defined in the Total DDL) or if *total-element-name* does not begin with the 4-character name of the file for which it is defined.

idms-name

Specifies the CA-IDMS/DB name to be used in place of the name supplied by the generator. CA-IDMS/DB element names must be from 1 to 32 characters long, and CA-IDMS/DB record and set names must be from 1 to 16 characters long.

Example: The following is an example of the SYNONYMS control statement.

```
SYNONYMS
  CUST          CUSTOMER
  CUSTCTRL      CUSTOMER-KEY
  CUOR          CUSTOMER-ORDER
  ORNU-CO-CUOR  CUST-TO-ORDER
  SAMENAME OF DARE  SAME-NAME-OF-DARE
```

2.2.4 AREA statement

The AREA statement allows you to specify the CA-IDMS/DB database area(s) to which CA-IDMS/DB records replacing Total files are assigned.

When generating schema record descriptions, the generator assigns each record type to the first area identified in the SCHEMA statement. The AREA statement permits you to override this default by naming specific Total files and the database areas to which corresponding CA-IDMS/DB records are assigned.

Syntax: The following is the syntax for the AREA statement. You must specify the keyword AREA in column 1, *idms-area-name* in column 8, and *total-file-name* in column 12. The keyword and each variable must occupy an entire line.

```
➤— AREA —————➤
      |
      |  ┌───┐
      |  |   |
      |  |idms-area-name|total-file-name|—————➤
```

Parameters

AREA

Required keyword; must occupy a line by itself and begin in column 1.

idms-area-name

Specifies the name of a CA-IDMS/DB database area to which CA-IDMS/DB records corresponding to subsequently named Total files are assigned. *Idms-area-name* must be from 1 to 16 characters long, must occupy a line by itself, and must begin in column 8. Additionally, *idms-area-name* must be followed by at least one Total file name and must be named in the SCHEMA statement.

total-file-name

Specifies the name of a Total file. All occurrences of the CA-IDMS/DB record type that correspond to this Total file are assigned to the database area indicated by *idms-area-name*. *Total-file-name* must be the name of a Total file defined in the input Total DDL. *Total-file-name* can be repeated as required, however each entry must occupy a line by itself, and must begin in column 12.

Example: The following is an example of the AREA control statement.

```
AREA
      CUSTOMER-REGION
      CUST
      REMA
      CUOR
      ACRE
      INVE
      DARE
      ORNU
```

2.2.5 SCHEMA statement

The generator SCHEMA statement describes the schema, files, and areas for CA-IDMS/DB. This information is used to generate the CA-IDMS/DB DDL SCHEMA, FILE, and AREA statements.

►► For information on CA-IDMS logical DDL, refer to *CA-IDMS Database Administration*.

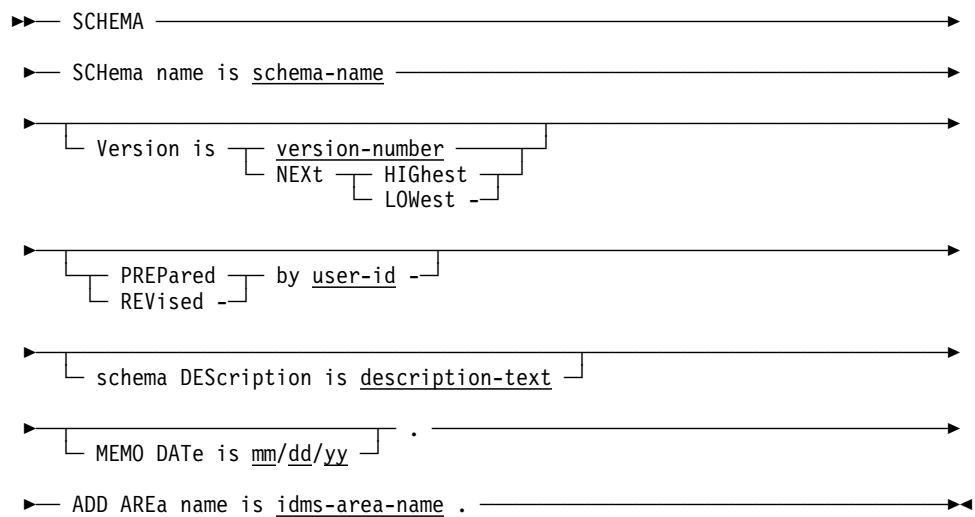
Composed of substatements The generator SCHEMA statement is composed of these substatements:

- SCHEMA — Names the schema and allows informational entries
- AREA — Names CA-IDMS/DB database areas and maps them into the named database files. To describe all areas in the CA-IDMS/DB database, you can add as many areas as you need in AREA substatements.

Using the information included in the SCHEMA statement and (if provided) in the SYNONYMS and AREA statements the generator produces a CA-IDMS/DB schema description that reflects the definition of the Total database defined in the input Total DDL. The schema that is produced includes complete definitions of the CA-IDMS/DB record types that correspond to the Total files and describes the CA-IDMS/DB sets that correspond to relationships among Total files.

The generator automatically generates the schema RECORD DESCRIPTION and SET DESCRIPTION components, converting Total file definitions to CA-IDMS/DB record descriptions and Total master file-variable file relationships to CA-IDMS/DB set descriptions. Any entries in the SYNONYMS control statement are applied to the appropriate record, element, and set descriptions. Any entries in the AREA statement are applied to the appropriate record descriptions.

Syntax: The following is the syntax for the SCHEMA statement. You must specify the keyword SCHEMA in column 1. All other clauses must begin between columns 8 and 11.



Parameters

SCHEMA

Required keyword; must occupy a line by itself and begin in column 1.

SCHEMA NAME IS schema-name

Specifies the name of the schema produced by the generator. *Schema-name* must be a 1- to 8-character value. The first character must be #, \$, @, or A through Z. The remaining characters can be #, \$, @, A through Z, 0 through 9, or the hyphen (except as the last character or following another hyphen).

VERSION

Qualifies the schema with a version number. Version numbers must fall within the range 1 through 9999, whether specified explicitly or specified in relation to existing versions.

version-number

Specifies an explicit version number and must be an unsigned integer in the range 1 through 9999.

NEXT HIGHEST

Specifies the highest version number assigned to *schema-name* plus one. For example, if versions 3, 5, and 8 of schema ETOTSCHM exist in the dictionary, the following statement would result in version 9 of ETOTSCHM being added to the dictionary:

```
SCHEMA NAME IS ETOTSCHM VERSION IS NEXT HIGHEST.
```

Note: If NEXT is specified without HIGHEST or LOWEST, the schema compiler assumes NEXT HIGHEST.

NEXT LOWEST

Specifies the lowest version number assigned to *schema-name* minus one. For example, if versions 3, 5, and 8 of schema ETOTSCHM exist in the dictionary, the following statement would result in version 2 of ETOTSCHM being added to the dictionary:

```
SCHEMA NAME IS ETOTSCHM VERSION IS NEXT LOWEST.
```


Note: If NEXT is specified without HIGHEST or LOWEST, the schema compiler assumes NEXT HIGHEST.

PREPARED/REVISED BY user-name

Identifies the schema author. *User-name* can be any 1- to 32-character value. If the value includes spaces or delimiters, it must be enclosed in quotation marks. PREPARED/REVISED BY is informational only.

SCHEMA DESCRIPTION IS description-text

Specifies remarks concerning the schema. *Description-text* is a 1- to 40-character alphanumeric value. If it contains spaces or delimiters, it must be enclosed in quotation marks. SCHEMA DESCRIPTION is informational only.

MEMO DATE IS mm/dd/yy

Specifies the date on which the schema was created. MEMO DATE is informational only.

ADD AREA NAME IS idms-area-name

Specifies the name of a CA-IDMS/DB database area. *Idms-area-name* must be a 1- to 16-character value. The characters can be #, \$, @, A through Z, 0 through 9, or the hyphen (except as the first or last character or following another hyphen). At least one alphabetic or international symbol (#, \$, @) is required.

Idms-area-name must not be the same as the schema name or the name of any other component (including synonyms) within the schema.

Idms-area-name is copied into DML programs and the DMCL, so do not use a keyword known to either the DMCL or the DML compiler. For a list of these keywords, refer to *CA-IDMS Database Administration*.

Example: The following is an example of the SCHEMA control statement.

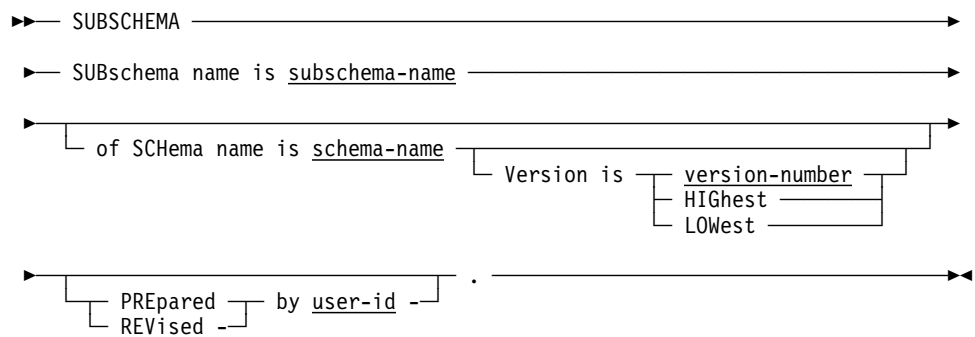
```
SCHEMA
    SCHEMA NAME IS ETOTSCHM VERSION IS 1
    MEMO DATE IS 10/19/91.

    ADD
    AREA NAME IS CUSTOMER-REGION.
```

2.2.6 SUBSCHEMA statement

The SUBSCHEMA statement names the subschema, associates it with a schema, and permits informational entries. The generator copies all area, record, and set descriptions included in the schema, providing access to the entire CA-IDMS/DB database as defined in the schema. The generator also appends VALIDATE and GENERATE statements to produce a subschema load module. The subschema load module thus created is used by the runtime interface to satisfy database requests issued from Total application programs.

Syntax: The following is the syntax for the SUBSCHEMA statement. All clauses must begin between columns 8 and 11, with the exception of the OF SCHEMA NAME clause, which must be coded between columns 12 and 72.



Parameters

SUBSCHEMA

Required keyword; must occupy a line by itself and begin in column 1.

SUBSCHEMA NAME IS subschemaname

Identifies the source description of the subschema to the data dictionary.

Subschemaname must be a 1- to 8-character value. The first character must be #, \$, @, or A through Z. The remaining characters can be #, \$, @, A through Z, 0 through 9, or the hyphen (except as the last character or following another hyphen). *Subschemaname* also becomes the CSECT name of the object subschema. If the object subschema subsequently is link edited alone for dynamic loading, *subschemaname* must be the load library member name.

OF SCHEMA NAME schema-name

Associates the named subschema with the generator-created schema. *Schema-name* must be the name of the schema identified in the SCHEMA statement.

VERSION version-number

Qualifies *schema-name* with a version number. *Version-number* must be an unsigned integer between 1 and 9999.

PREPARED/REVISED BY user-id

Identifies the user who is preparing the subschema. PREPARED/REVISED BY is informational only.

Example: The following is an example of the SUBSCHEMA control statement.

```

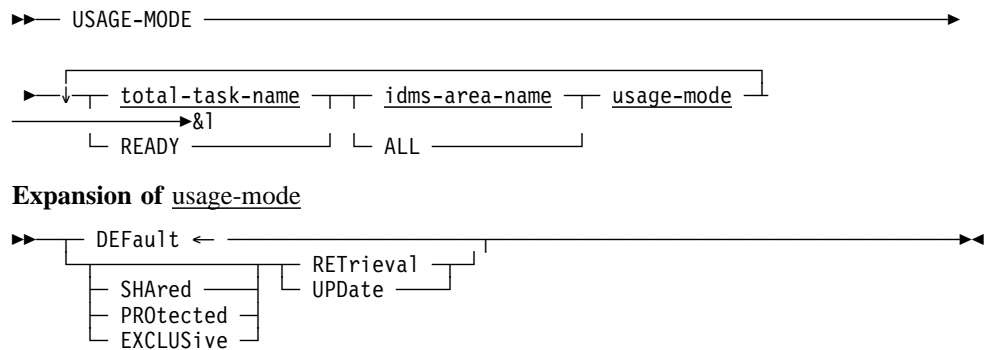
SUBSCHEMA
SUBSCHEMA NAME IS ETOTSUBS OF SCHEMA ETOTSCHM.

```

2.2.7 USAGE-MODE statement

Before performing database services for a given Total application program, the runtime interface readies each area defined in the generated subschema in shared update usage mode. The optional USAGE-MODE statement allows you to override the default usage mode for areas used by the Total task associated with the application program.

Syntax: Each clause of this statement (after the keyword USAGE-MODE) must occupy a line by itself and must begin in column 8. Include as many clauses as needed.



Expansion of usage-mode

Parameters

USAGE-MODE

Required keyword; must occupy a line by itself and begin in column 1.

total-task-name/READY

Identifies the Total tasks that access the named area in the specified usage mode.

total-task-name

Specifies the name of the Total task as it exists in the SINON command issued by the Total application program.

Total-task-name USAGE-MODE statements must precede any READY USAGE-MODE statements.

READY

Specifies that all Total tasks that do not match *total-task-name* are to be readied in the named usage mode.

Total-task-name USAGE-MODE statements must precede any READY USAGE-MODE statements.

idms-area-name/ALL

Identifies the CA-IDMS/DB areas affected by each USAGE-MODE statement.

idms-area-name

Identifies the CA-IDMS/DB database area known to the subschema used by the runtime interface.

ALL

Specifies that the runtime interface is to issue a READY ALL DML verb, indicating to CA-IDMS/DB that all areas are to be accessed in the specified usage mode. If you specify *usage-mode* (parameter explained below) as DEFAULT, all areas are readied in the usage mode specified in the CA-IDMS/DB subschema.

usage-mode

Identifies a valid CA-IDMS/DB usage mode.

DEFault

Specifies that all areas are to be readied in the usage mode specified in the CA-IDMS/DB subschema.

Note: DEFAULT is valid only if ALL is specified; it is not used with *idms-area-name*.

SHAreD

Allows other run units executing concurrently under the CA-IDMS/DB central version (CV) either to ready the area in shared update or shared retrieval mode (SHARED UPDATE), or to ready the area in shared update, shared retrieval, protected update, or protected retrieval mode (SHARED RETRIEVAL).

PROtected

Prevents concurrent update of the area by run units executing under the same central version. Once a run unit has readied an area with the PROTECTED option, no other run unit can ready that area in any UPDATE usage mode until the first run unit releases it. A run unit cannot ready an area with the PROTECTED option if another run unit has readied the area in UPDATE usage mode or with the EXCLUSIVE option.

EXCLUSive

Prevents concurrent use of the area by any other run unit executing under the CA-IDMS/DB central version. Once a run unit has readied an area with the EXCLUSIVE option, no other run unit can ready that area in any usage mode until the first run unit releases it.

RETrieval

Opens the area for retrieval only and allows other concurrently executing run units to open the same area in any usage mode other than one that is exclusive.

UPDate

Opens the area for both retrieval and update and allows other concurrently executing run units to open the same area in any usage mode other than one that is exclusive or protected.

Usage

Mutually exclusive clauses: Do not use READY ALL and READY *idms-area-name* clauses in the same USAGE-MODE statement.

Do not use *total-task-name* ALL and *total-task-name idms-area-name* clauses in the same USAGE-MODE statement.

READY ALL SHARED UPDATE: The USAGE-MODE statement READY ALL SHARED UPDATE is the same as having no USAGE-MODE statements.

Bypassing DML READY verbs: The USAGE-MODE statement READY ALL DEFAULT causes the runtime interface to bypass issuing any DML READY verbs. The areas needed for processing will be readied as specified in the subschema.

Central version parameters: The SHARED, PROTECTED, and EXCLUSIVE options apply only to programs running under the CA-IDMS/DB central version.

Examples

Example 1: Suppose there are two tasks associated with an ETOTTBL and you wish to open the database areas associated with each task in different usage modes. USERPRG1 uses CA-IDMS-ADBEA1 and CA-IDMS-ADBEA2; USERPRG2 uses CA-IDMS-ADBEA2 and CA-IDMS-ADBEA3. Sample statements are listed below:

USAGE-MODE		
USERPRG1	CA-IDMS-ADBEA1	RETRIEVAL
USERPRG1	CA-IDMS-ADBEA2	PROTECTED UPDATE
USERPRG2	CA-IDMS-ADBEA3	EXCLUSIVE UPDATE

The areas for USERPRG1 are readied as specified in the USAGE-MODE table; for USERPRG2, CA-IDMS-ADBEA3 is readied as specified and CA-IDMS-ADBEA2 defaults to SHARED UPDATE. All areas in the subschema not readied by specific USAGE-MODE statements will be readied in SHARED UPDATE usage mode.

Example 2: Suppose you have many CA-IDMS/DB areas in the subschema associated with an ETOTTBL. The task does not require access to all of the areas, so only required areas are readied. Sample statements are listed below:

USAGE-MODE		
USERPRG1	CA-IDMS-ADBEA1	RETRIEVAL
USERPRG1	CA-IDMS-ADBEA2	PROTECTED UPDATE
USERPRG2	CA-IDMS-ADBEA2	SHARED UPDATE
USERPRG2	CA-IDMS-ADBEA3	EXCLUSIVE UPDATE
READY	ALL	DEFAULT

The results are the same as they are for example 1, except that any other areas in the CA-IDMS/DB subschema will not be readied. Any task other than USERPRG1 or USERPRG2 will have no DML READY verb generated. The last statement in the example causes any other tasks to use the default ready mode as specified in the subschema.

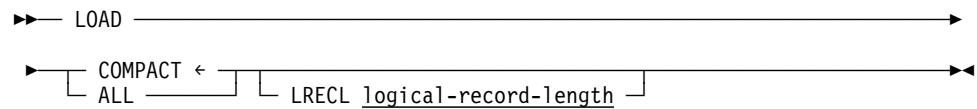
2.2.8 LOAD statement

If the unloaded Total data contains master file control keys and linkpath fields, use the LOAD statement to tell the loader so that these fields can be ignored when the corresponding CA-IDMS/DB database is loaded. If master file control keys and linkpath fields are not present, the LOAD statement is optional.

Unloaded Total data includes master file control keys and linkpath fields only if a means other than the CSITULOD utility is used to unload the Total database.

Additionally, the LOAD statement includes a parameter to specify the logical record length of the unloaded Total data if this value is other than 120 bytes.

Syntax



Parameters

LOAD

Required keyword; must occupy a line by itself and begin in column 1.

COMPACT/ALL

Specifies whether the unloaded Total data is to contain master file control keys and linkpath fields. Begin this clause in column 8.

COMPACT

Specifies that master records, standard variable records, and coded variable records unloaded from the Total database contain the file name in the first four bytes of the record, followed immediately by all data items. Linkpath fields are not present in an unloaded Total record, nor is the master file control key defined in Total DDL for master files.

COMPACT is the default. It is informational only and specifies that the unloaded Total data will not contain these fields.

ALL

Specifies that unloaded Total master records, standard variable records, and coded variable records contain the information shown in the following table.

Use ALL if you are using the CSITULOD utility to unload the Total database (specify COMPACT in exceptional cases only). If you use means other than CSITULOAD to unload the Total database, the contents of the unloaded Total records must conform to one of the options provided by this clause.

Record(s)	Contents
Master	The first four bytes of each unloaded master record contain the master file name, followed by four bytes containing the internal reference point, followed by the master file control key, followed by the contents of each linkpath field (eight bytes per linkpath) defined for the master file, followed by all data items defined for the record.
Standard variable	The first four bytes of each unloaded standard variable record contain the standard variable file name, followed by the contents of the principal linkpath key for the associated master file, followed by all data items defined for the record.
Coded variable	The first four bytes of each unloaded coded variable record contain the coded variable file name, followed by two bytes containing the record code, followed by the contents of the principal linkpath key for the associated master file, followed by all data items defined for the record.

LRECL logical-record-length

Indicates the largest logical record length of unloaded Total data if this length is other than 120 bytes. Do not include LRECL if the logical record length is 120. You must specify the logical record length of the largest record if you are using variable-length records. Begin this clause in column 8.

Example: The following is an example of the LOAD control statement.

```
LOAD  
      ALL
```


Chapter 3. Database Conversion Phase

3.1 Steps of the database conversion phase	3-3
3.2 Step 1: Transparency generator (ETOTMAIN)	3-4
3.3 Step 2: Schema preparation	3-13
3.4 Step 3: Physical database preparation	3-14
3.5 Step 4: Subschema preparation	3-16
3.6 Step 5: CA-IDMS/DB database initialization	3-18
3.7 Step 6: Total database unload utility	3-19
3.8 Step 7: Transparency loader (ETOTLOAD)	3-20

3.1 Steps of the database conversion phase

The process for converting a Total database to a CA-IDMS/DB database involves the following steps:

1. Generate the data definitions and load program using the transparency generator ETOTMAIN
2. Compile the generated schema, using the schema compiler IDMSCHEM; this stores the schema description in the dictionary
3. Prepare the physical database description by:
 - a. Submitting physical DDL statements to the CA-IDMS Command Facility (IDMSBCF)
 - b. Generating database name table and DMCL load modules
 - c. Punching the database name table and DMCL load modules using the PUNCH utility statement
 - d. Linking the resulting object modules to a load library
4. Prepare the subschema by:
 - a. Compiling the generated subschema using the subschema compiler IDMSUBSC
 - b. Punching the subschema load module
 - c. Linking the subschema to a load library
5. Initialize the CA-IDMS/DB database using the FORMAT utility statement
6. Unload the Total database using CSITULOD
7. Load the CA-IDMS/DB database using the customized transparency loader ETOTLOAD or a user-written load program

These conversion steps are discussed in this chapter.

►► For OS/390 JCL, see Appendix F, “OS/390 Job Control Language” on page F-1.
For VSE/ESA JCL, see Appendix G, “VSE/ESA Job Control Language” on page G-1.

3.2 Step 1: Transparency generator (ETOTMAIN)

The first step in transparency conversion of a Total database to a CA-IDMS/DB database is to execute the transparency generator ETOTMAIN.

Input to ETOTMAIN: You provide the following input to ETOTMAIN, in this order:

- Total DDL that defines the database being converted
- Control statements that describe the CA-IDMS/DB database environment
- ETOTTBL skeleton
- ETOTLOAD skeleton

Output from ETOTMAIN: Output from ETOTMAIN includes:

- Schema source code
- Subschema source code
- Customized runtime control table
- Customized loader

►► For information on messages issued by ETOTMAIN, see Appendix B, “Generator Messages” on page B-1.

Examples of input and output

Input — Total DDL: The following is an example of Total DDL statements input to the transparency generator (ETOTMAIN).

```
BEGIN-DATA-BASE-GENERATION:
DATA-BASE-NAME=CUORDABA      16:28:54 07/18/99   08/18/99

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=CUST
MASTER-DATA:
CUSTROOT=8
CUSTCTRL=6
CUSTLKCO=8
CUSTLKAR=8
CUSTLKRE=8
CUSTNAME=30
CUSTADDR=30
CUSTCIST=20
*FILLER*=5
SAMENAME=5
END-DATA:
LOGICAL-RECORD-LENGTH=128
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=ORNU
MASTER-DATA:
ORNURoot=8
ORNUCTRL=12
ORNULKCO=8
.00.ORN0001=5
.00.ORN0002=0
. .01.ORN0003=2
. .01.ORN0004=0
. .02.ORN0005=7
. .02.ORN0006=6
. .03.ORN0007=1
. .03.ORN0008=5
. .02.ORN0009=0
. .03.ORN0010=2
. .03.ORN0011=4
END-DATA:
LOGICAL-RECORD-LENGTH=49
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=DARE
MASTER-DATA:
DARERoot=8
DARECTRL=4
DARELKDR=8
DARELKDS=8
DARELKDN=8
DARELKDG=8
SAMENAME=6
END-DATA:
LOGICAL-RECORD-LENGTH=50
END-MASTER-DATA-SET:
```

```
BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=INVE
MASTER-DATA:
INVEROOT=8
INVECTRL=6
INVELKCO=8
INVELKRE=8
INVEITDE=30
INVEITCO=5
INVEITPR=5
INVEQTOH=4
INVEQT00=4
END-DATA:
LOGICAL-RECORD-LENGTH=78
END-MASTER-DATA-SET:

BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=CUOR
BASE-DATA:
CUORCODE=2
CUORORNU=12=ORNUCTRL
ORNULKCO=8
CUORLINU=2
CUORDATA=55
RECORD-CODE=HD
CUORDRDR=4=DARECTRL
DARELKDR=8
CUORDSDS=4=DARECTRL
DARELKDS=8
CUORCUST=6=CUSTCTRL
CUSTLKCO=8
CUORTOVA=5
CUORTOIT=2
CUORTERM=10
RECORD-CODE=IT
CUORINVE=6=INVECTRL
INVELKCO=8
CUORQURE=4
CUORPRIC=5
CUORWEIG=4
RECORD-CODE=CM
CUORCOMM=55
RECORD-CODE=JG
CUORJGDR=4
DARELKDR=8=CUORJGDR
CUORJGDS=4
DARELKDS=8=CUORJGDS
END-DATA:
LOGICAL-RECORD-LENGTH=79
END-VARIABLE-ENTRY-DATA-SET:
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=ACRE
BASE-DATA:
ACRECODE=2
ACRECUST=6=CUSTCTRL
CUSTLKAR=8
ACRESENB=2
ACREDATA=45
RECORD-CODE=BL
ACREDNDN=4=DARECTRL
DARELKDN=8
ACREDGDG=4=DARECTRL
DARELKDG=8
ACREINN=6
ACRENEAM=5
ACREGRAM=5
ACREMPA=5
RECORD-CODE=CK
ACRECKNB=6
ACRENUMB=6
ACREAMOU=5
END-DATA:
LOGICAL-RECORD-LENGTH=63
END-VARIABLE-ENTRY-DATA-SET:
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=REMA
BASE-DATA:
REMACUST=6=CUSTCTRL
CUSTLKRE=8
REMAINVE=6=INVECTRL
INVELKRE=8
REMASENB=2
REMAEXT=78
END-DATA:
LOGICAL-RECORD-LENGTH=108
END-VARIABLE-ENTRY-DATA-SET:
```

```
END-DATA-BASE-GENERATION:
```

Output — Schema source: The following is an example of schema source statements generated by the transparency generator, ETOTMAIN. This output is based on the previous Total DDL example input.

3.2 Step 1: Transparency generator (ETOTMAIN)

```
ADD
SCHEMA NAME IS ETOTSCHM VERSION IS 1.

ADD
AREA NAME IS CUSTOMER-REGION.

ADD RECORD NAME IS CUSTOMER
RECORD ID IS 101
LOCATION MODE IS CALC USING CUSTOMER-KEY
                                DUPLICATES ARE NOT ALLOWED
                                WITHIN AREA CUSTOMER-REGION.

02 CUSTOMER-KEY                  PIC X(6).
02 CUSTNAME                      PIC X(30).
02 CUSTADDR                      PIC X(30).
02 CUSTCIST                      PIC X(20).
02 FILLER                       PIC X(5).
02 SAMENAME                     PIC X(5).

ADD RECORD NAME IS ORNU
RECORD ID IS 102
LOCATION MODE IS CALC USING ORNUCTRL
                                DUPLICATES ARE NOT ALLOWED
                                WITHIN AREA CUSTOMER-REGION.

02 ORNUCTRL                      PIC X(12).
02 ORNU0001                     PIC X(5).
02 ORNU0002.
03 ORNU0003                     PIC X(2).
03 ORNU0004.
04 ORNU0005                     PIC X(7).
04 ORNU0006.
05 ORNU0007                     PIC X(1).
05 ORNU0008                     PIC X(5).
04 ORNU0009.
05 ORNU0010                     PIC X(2).
05 ORNU0011                     PIC X(4).
02 FILLER                       PIC X(2).

ADD RECORD NAME IS DARE
RECORD ID IS 103
LOCATION MODE IS CALC USING DARECTRL
                                DUPLICATES ARE NOT ALLOWED
                                WITHIN AREA CUSTOMER-REGION.

02 DARECTRL                      PIC X(4).
02 SAME-NAME-OF-DARE            PIC X(6).
02 FILLER                       PIC X(2).

ADD RECORD NAME IS INVE
RECORD ID IS 104
LOCATION MODE IS CALC USING INVECTRL
                                DUPLICATES ARE NOT ALLOWED
                                WITHIN AREA CUSTOMER-REGION.
```



```

02  INVECTRL          PIC X(6).
02  INVEITDE          PIC X(30).
02  INVEITCO          PIC X(5).
02  INVEITPR          PIC X(5).

02  INVEQTOH          PIC X(4).
02  INVEQTOO          PIC X(4).
02  FILLER            PIC X(2).

```

ADD RECORD NAME IS CUSTOMER-ORDER

```

RECORD ID IS 105
LOCATION MODE IS VIA CUST-TO-ORDER SET
                    WITHIN AREA CUSTOMER-REGION.

```

```

02  CUORCODE          PIC X(2).
02  CUORORNU          PIC X(12).
02  CUORLINU          PIC X(2).
02  CUORDATA          PIC X(55).
02  CUOR-RECORD-HD

                                REDEFINES CUORDATA.
03  CUORDRDR          PIC X(4).
03  CUORDSDS          PIC X(4).
03  CUORCUST          PIC X(6).
03  CUORTOVA          PIC X(5).
03  CUORTOIT          PIC X(2).
03  CUORTERM          PIC X(10).
03  FILLER            PIC X(24).
02  CUOR-RECORD-IT

                                REDEFINES CUORDATA.
03  CUORINVE          PIC X(6).
03  CUORQURE          PIC X(4).
03  CUORPRIC          PIC X(5).
03  CUORWEIG          PIC X(4).
03  FILLER            PIC X(36).
02  CUOR-RECORD-CM

                                REDEFINES CUORDATA.
03  CUORCOMM          PIC X(55).
02  CUOR-RECORD-JG

                                REDEFINES CUORDATA.
03  CUORJGDR          PIC X(4).
03  CUORJGDS          PIC X(4).
03  FILLER            PIC X(47).
02  FILLER            PIC X(1).

```

```

ADD RECORD NAME IS ACRE
RECORD ID IS 106
LOCATION MODE IS VIA CUST-AR-ACRE SET
                    WITHIN AREA CUSTOMER-REGION.

```

```

02  ACRECODE          PIC X(2).
02  ACRECUST          PIC X(6).

```

```

02  ACRESNB                      PIC X(2).
02  ACREDATA                     PIC X(45).
02  ACRE-RECORD-BL
                                REDEFINES ACREDATA.
    03  ACREDNDN                 PIC X(4).
    03  ACREDGDG                 PIC X(4).
    03  ACREINNB                 PIC X(6).
    03  ACRENEAM                 PIC X(5).
    03  ACREGRAM                 PIC X(5).
    03  ACREAMPA                 PIC X(5).
    03  FILLER                   PIC X(16).
02  ACRE-RECORD-CK
                                REDEFINES ACREDATA.
    03  ACRECKNB                 PIC X(6).
    03  ACRENUMB                 PIC X(6).
    03  ACREAMOU                 PIC X(5).
    03  FILLER                   PIC X(28).
02  FILLER                      PIC X(1).

ADD RECORD NAME IS REMA
RECORD ID IS 107
LOCATION MODE IS VIA CUST-RE-REMA SET
                                WITHIN AREA CUSTOMER-REGION.

02  REMACUST                     PIC X(6).
02  REMAINVE                     PIC X(6).
02  REMASENB                     PIC X(2).
02  REMATEXT                     PIC X(78).

ADD SET NAME IS CUST-TO-ORDER
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS ORNU
                                LINKED TO PRIOR
                                NEXT DBKEY POSITION IS 1
                                PRIOR DBKEY POSITION IS 2

MEMBER IS CUSTOMER-ORDER
                                NEXT DBKEY POSITION IS 1
                                PRIOR DBKEY POSITION IS 2
                                MANDATORY AUTOMATIC.

ADD SET NAME IS DARE-DR-CUOR
ORDER IS PRIOR
MODE IS CHAIN
OWNER IS DARE
                                LINKED TO PRIOR
                                NEXT DBKEY POSITION IS 1
                                PRIOR DBKEY POSITION IS 2

MEMBER IS CUSTOMER-ORDER
                                NEXT DBKEY POSITION IS 3
                                PRIOR DBKEY POSITION IS 4
                                MANDATORY MANUAL.

ADD SET NAME IS DARE-DS-CUOR
ORDER IS PRIOR
MODE IS CHAIN
                                LINKED TO PRIOR

```

OWNER IS DARE	NEXT DBKEY POSITION IS 3 PRIOR DBKEY POSITION IS 4
MEMBER IS CUSTOMER-ORDER	NEXT DBKEY POSITION IS 5 PRIOR DBKEY POSITION IS 6 MANDATORY MANUAL.
ADD SET NAME IS CUST-CO-CUOR ORDER IS PRIOR MODE IS CHAIN OWNER IS CUSTOMER	LINKED TO PRIOR NEXT DBKEY POSITION IS 1 PRIOR DBKEY POSITION IS 2
MEMBER IS CUSTOMER-ORDER	NEXT DBKEY POSITION IS 7 PRIOR DBKEY POSITION IS 8 MANDATORY MANUAL.
ADD SET NAME IS INVE-CO-CUOR ORDER IS PRIOR MODE IS CHAIN OWNER IS INVE	LINKED TO PRIOR NEXT DBKEY POSITION IS 1 PRIOR DBKEY POSITION IS 2
MEMBER IS CUSTOMER-ORDER	NEXT DBKEY POSITION IS 9 PRIOR DBKEY POSITION IS 10 MANDATORY MANUAL.
ADD SET NAME IS CUST-AR-ACRE ORDER IS PRIOR MODE IS CHAIN OWNER IS CUSTOMER	LINKED TO PRIOR NEXT DBKEY POSITION IS 3 PRIOR DBKEY POSITION IS 4
MEMBER IS ACRE	NEXT DBKEY POSITION IS 1 PRIOR DBKEY POSITION IS 2 MANDATORY AUTOMATIC.
ADD SET NAME IS DARE-DN-ACRE ORDER IS PRIOR MODE IS CHAIN OWNER IS DARE	LINKED TO PRIOR NEXT DBKEY POSITION IS 5 PRIOR DBKEY POSITION IS 6
MEMBER IS ACRE	NEXT DBKEY POSITION IS 3 PRIOR DBKEY POSITION IS 4 MANDATORY MANUAL.
ADD SET NAME IS DARE-DG-ACRE ORDER IS PRIOR MODE IS CHAIN OWNER IS DARE	LINKED TO PRIOR NEXT DBKEY POSITION IS 7 PRIOR DBKEY POSITION IS 8

MEMBER IS ACRE	NEXT DBKEY POSITION IS 5 PRIOR DBKEY POSITION IS 6 MANDATORY MANUAL.
ADD SET NAME IS CUST-RE-REMA ORDER IS PRIOR MODE IS CHAIN OWNER IS CUSTOMER	LINKED TO PRIOR NEXT DBKEY POSITION IS 5 PRIOR DBKEY POSITION IS 6
MEMBER IS REMA	NEXT DBKEY POSITION IS 1 PRIOR DBKEY POSITION IS 2 MANDATORY AUTOMATIC.
ADD SET NAME IS INVE-RE-REMA ORDER IS PRIOR MODE IS CHAIN OWNER IS INVE	LINKED TO PRIOR NEXT DBKEY POSITION IS 3 PRIOR DBKEY POSITION IS 4
MEMBER IS REMA	NEXT DBKEY POSITION IS 3 PRIOR DBKEY POSITION IS 4 MANDATORY MANUAL.
VALIDATE.	

Output — Subschema source: The following is a sample of a subschema that is generated by the transparency generator, ETOTMAIN. This output is based on the previous Total DDL example input.

```
ADD
SUBSCHEMA NAME IS ETOTSUBS OF SCHEMA NAME ETOTSCHM.

  ADD AREA CUSTOMER-REGION.

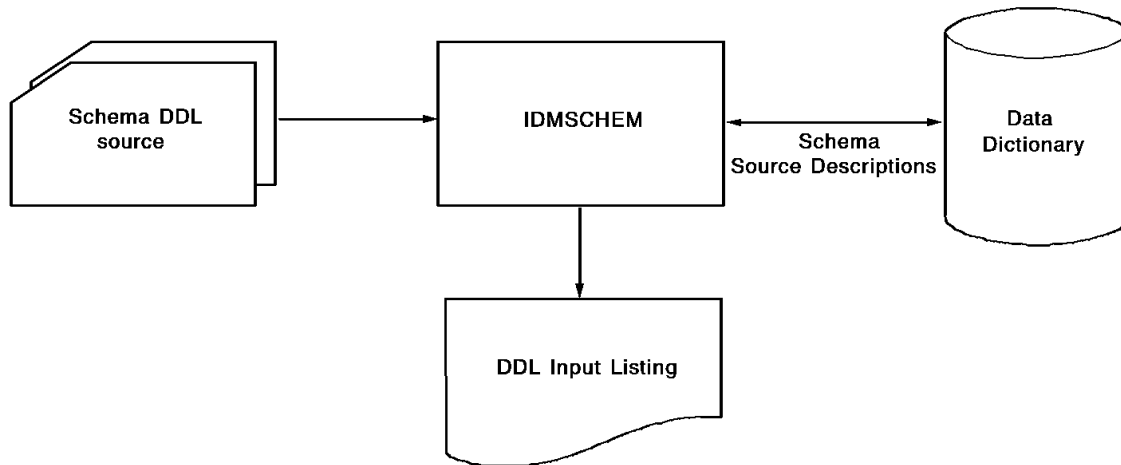
  ADD RECORD CUSTOMER.
  ADD RECORD ORNU.
  ADD RECORD DARE.
  ADD RECORD INVE.
  ADD RECORD CUSTOMER-ORDER.
  ADD RECORD ACRE.
  ADD RECORD REMA.

  ADD SET CUST-TO-ORDER.
  ADD SET DARE-DR-CUOR.
  ADD SET DARE-DS-CUOR.
  ADD SET CUST-CO-CUOR.
  ADD SET INVE-CO-CUOR.
  ADD SET CUST-AR-ACRE.
  ADD SET DARE-DN-ACRE.
  ADD SET DARE-DG-ACRE.
  ADD SET CUST-RE-REMA.
  ADD SET INVE-RE-REMA.
  VALIDATE.
  GENERATE.
```

3.3 Step 2: Schema preparation

To compile the schema created by the transparency generator, execute the program IDMSCHEM.

Schema compiler execution: The following figure shows the execution of the schema compiler.



Input to IDMSCHEM: The input to IDMSCHEM is the source schema DDL output by the generator.

Output from IDMSCHEM: Output from IDMSCHEM is as follows:

- A description of the schema stored in the dictionary
- A schema DDL input listing, including embedded warning and error messages issued by IDMSCHEM

►► For descriptions messages issued by IDMSCHEM, refer to *CA-IDMS Messages and Codes*. If you want to use the extended options of the schema compiler, refer to *CA-IDMS Database Administration*.

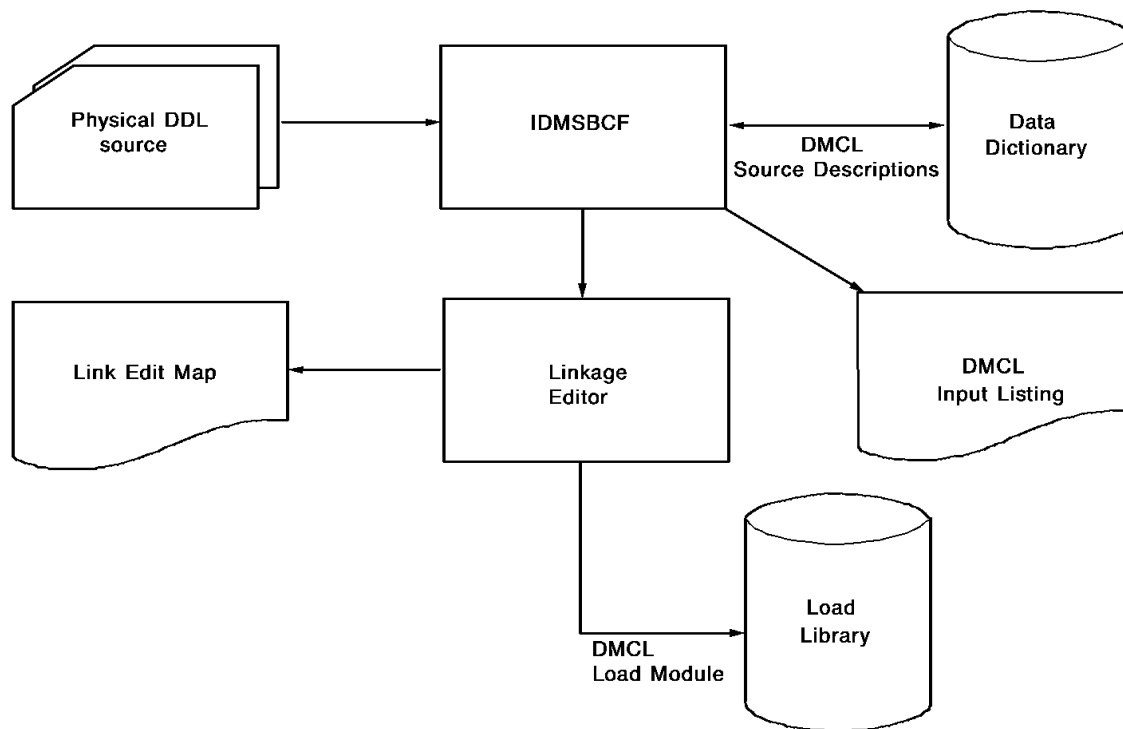
3.4 Step 3: Physical database preparation

To prepare the physical database, perform these steps:

1. Submit DDL statements that describe the physical database using the CA-IDMS Command Facility (IDMSBCF); physical DDL statements include the SEGMENT, DBNAME TABLE, and DMCL statements; include a GENERATE clause to generate database name table and DMCL load modules
2. Punch the database name table and DMCL load modules using the PUNCH utility statement
3. Link the resulting object modules to a load library
4. Identify the DMCL to the runtime system:
 - Central version — specify the global DMCL name in the #DCPARM macro which defines system startup parameters; incorporate the Total DMCL statements into the global DMCL using the MODIFY DMCL statement
 - Local mode — specify the DMCL name in the SYSIDMS parameter file

►► For information on physical DDL statements and SYSIDMS, refer to *CA-IDMS Database Administration*. For information on the CA-IDMS Command Facility, refer to *CA-IDMS Command Facility*. For information on #DCPARM, refer to *CA-IDMS System Operations*.

IDMSBCF execution: The following figure shows execution of IDMSBCF, the program you use to compile physical DDL.



Input to IDMSBCF: As input to IDMSBCF, you use the DDL statements that describe these physical components of the database:

- Segments
- Database name table (dbname table)
- DMCL

Output from IDMSBCF: Output from IDMSBCF is as follows:

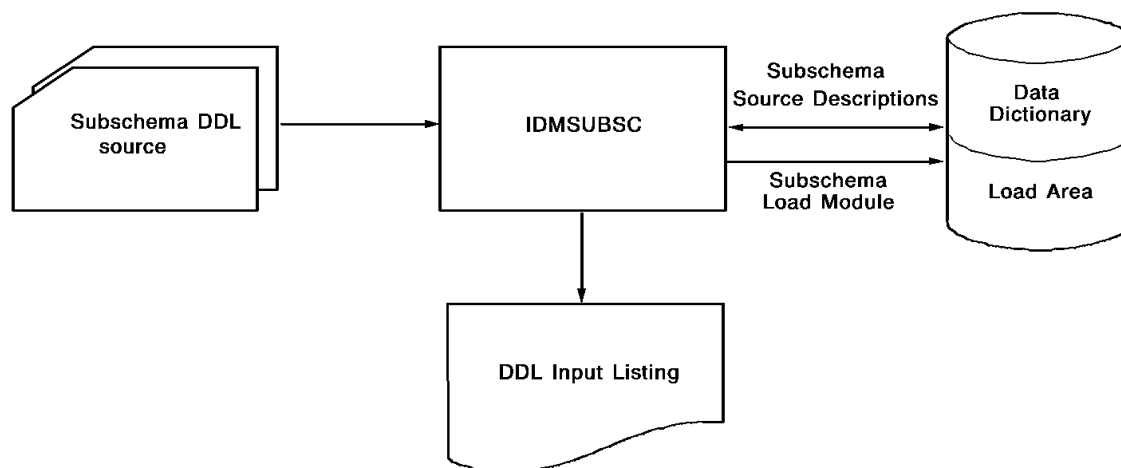
- DMCL and database name table load modules
- A physical DDL input listing, including embedded warning and error messages issued by IDMSBCF (for a description of these messages, refer to *CA-IDMS Messages and Codes*)

3.5 Step 4: Subschema preparation

To prepare the subschema, you perform these steps:

1. Compile the subschema using IDMSUBSC
2. Punch a subschema load module
3. Link edit the load module to a load library

IDMSUBSC execution: The following figure shows the execution of the subschema compiler IDMSUBSC.



Input to IDMSUBSC: Input to IDMSUBSC is the subschema DDL created by the transparency generator.

Output from IDMSUBSC: Output from IDMSUBSC is:

- A source description of the subschema stored in the dictionary, including embedded warning and error messages issued by IDMSUBSC
- A subschema load module stored in the dictionary load area (DDLDCLOD), if the source input contains a GENERATE statement; the subschema in the dictionary load area is used by applications running under the central version

►► For information on IDMSUBSC messages, refer to *CA-IDMS Messages and Codes*.

Punching the subschema load module: To punch the subschema load module, you submit a PUNCH LOAD MODULE statement to the DDDL compiler (IDMSDDDL).

Output from IDMSDDDL is an object deck. VSE/ESA users must be sure to save this output for use in step 7 of this conversion.

Linking the subschema load module: Input to the linkage editor is the object deck produced by IDMSDDDL.

Output from the linkage editor is a load module in the load library. The subschema in the load library is used by applications running in local mode and the transparency loader.

3.6 Step 5: CA-IDMS/DB database initialization

To initialize the CA-IDMS/DB database (format database pages), you submit the FORMAT utility statement through the CA-IDMS Command Facility (IDMSBCF).

►► For information on FORMAT, refer to *CA-IDMS Utilities*. For information on IDMSBCF, refer to *CA-IDMS Command Facility*.

3.7 Step 6: Total database unload utility

To unload the Total database, execute the Cincom utility, CSITULOD. Specify the appropriate parameters to preserve all pointers and unload all elements. Refer to the appropriate Total documentation for instructions on executing this utility.

If a means other than the CSITULOD utility is used to unload the Total database, ensure that the logical order of the database is preserved. Additionally, ensure that the content of all unloaded Total records conforms to that specified in the LOAD statement input to the transparency generator (see Chapter 2, “Data Description Phase” on page 2-1).

3.8 Step 7: Transparency loader (ETOTLOAD)

Compile, link edit, and execute the transparency customized loader, ETOTLOAD, or your own load program to load data into the CA-IDMS/DB database. You must run the loader in local mode.

►► For an explanation of the messages provided by the transparency loader, see Appendix C, “Loader Messages” on page C-1.

The ETOTLOAD job stream consists of several steps.

Before compiling ETOTLOAD: Review the file description (FD) of the UNLOAD-FILE and edit as necessary before you compile ETOTLOAD.

VSE/ESA users should also review the COBOL INPUT-OUTPUT SECTION ASSIGN clause and edit it as necessary.

Before executing ETOTLOAD: Before executing ETOTLOAD you must change all set orders of PRIOR to NEXT. To do this, execute the schema compiler IDMSCHEM with the following input:

```
modify schema schema-name.
modify set set-name-1 order is next.
modify set set-name-2 order is next.
.
.
.
modify set set-name-n order is next.
validate.
regenerate all subschemas.
```

Modify the set order for all sets in all schemas. Punch and link the subschema load modules.

After executing ETOTLOAD: After executing the loader you must restore all set orders to PRIOR. To do this, execute the schema compiler IDMSCHEM with the following input:

```
modify schema schema-name.
modify set set-name-1 order is prior.
modify set set-name-2 order is prior.
.
.
.
modify set set-name-n order is prior.
validate.
regenerate all subschemas.
```

Modify the set order for all sets in all schemas. Punch and link the subschema load modules.

Chapter 4. Runtime Operations Phase

4.1 Runtime steps	4-3
-----------------------------	-----

4.1 Runtime steps

Before you begin: See the following appendixes in this manual for additional information:

- Appendix F, “OS/390 Job Control Language” on page F-1 for OS/390 JCL.
- Appendix G, “VSE/ESA Job Control Language” on page G-1 for VSE/ESA JCL.
- Appendix D, “Runtime Messages” on page D-1 for a discussion of the codes returned to the Total application program running in the transparency environment

Refer to *CA-IDMS System Operations* for detailed information about runtime operations.

What to do: To execute a Total application program with the CA-IDMS/Total Transparency runtime interface, perform the steps below.

Step 1: Assemble and link edit ETOTTBL output: Assemble and link edit ETOTTBL output generated by ETOTMAIN.

Perform this step once to store the control table in the load library for use with all Total application programs. It is not repeated for each Total application program that uses the runtime interface.

Step 2: Modify the Total application program: Modify the Total application program to remove any Total DML functions not supported by the runtime interface (see 1.6.2, “Total features supported” on page 1-15).

Step 3: Recompile the Total application program: Recompile the Total application program if it has been modified.

Step 4: Assemble the ETOTCINT macro: Assemble the ETOTCINT macro if CICS applications will use the transparency to access the CA-IDMS/DB database.

The syntax for the ETOTCINT macro used to generate the ETOTCINT module is shown below. You only need to assemble this macro once. Subsequently, it can be used with all Total CICS applications to access the CA-IDMS/DB database.

```

▶▶ ETOTCINT CWADISP= cics-cwa-displacement
▶
  ,NAME= ETOTCINT ←
         csect-name
▶
  ,TYPE= MACRO
         EXEC
▶
  ,DFHEI1= NO ←
           YES
▶▶

```

Parameters

CWADISP=cics-cwa-displacement

Specifies the displacement, in bytes, of the transparency work area within the CICS CWA area. Specify the same value given to the CWADISP operand when you assembled the CICSOPT macro to create your CICSOPTS module.

TYPE=EXEC/MACRO

Specifies whether the CICS command-level interface or the CICS non-command-level interface is being used. This parameter is required. EXEC indicates that the command-level interface is in use; MACRO indicates that the non-command-level interface is in use.

NAME=ETOTCINT/csect-name

Specifies the name of the generated module. The default is ETOTCINT. If specified, *csect-name* must be a 1- to 8-character alphanumeric value.

DFHEI1=NO/YES

Specifies how the CICS CSA is located. NO is the default. NO specifies that R13 back chaining to CICS CSA is to be used whenever possible. YES specifies a call to the DFHEI1 routine to locate the CICS CSA. Use YES for VS COBOL II, PL/I, and VSE/ESA command level applications.

Step 5: Create your CICS interface module: Refer to F.6, “Creating the CICS CA-IDMS/DB Interface (IDMSINTC)” on page F-11 for information on this topic.

Step 6: Link edit the compiled Total application: Link edit the compiled Total application program with the following transparency and CA-IDMS/DB modules:

- ETOTBINT (batch only) or ETOTCINT (CICS only) — The runtime interface module that passes translated database calls to CA-IDMS/DB. ETOTBINT is supplied as an object module with the transparency; ETOTCINT is supplied as a macro that must be assembled prior to runtime.
- ETOTTRAN (batch only) — The runtime interface module (supplied in object form) that translates Total database calls to CA-IDMS/DB database calls.
- IDMS (batch only) — The CA-IDMS/DB module that will receive database calls passed by ETOTBINT.
- IDMSOPTI (batch only) — The (optional) CA-IDMS/DB module that passes runtime information to the CA-IDMS/DB central version that will handle database calls issued by the runtime interface of the transparency. IDMSOPTI is used to indicate the mode of operation.

Step 7: Execute the Total application: Execute the Total application as per standard CA-IDMS/DB runtime procedures.

Appendix A. User Exits

A.1	DATBASBF and DATBASAF exits	A-3
A.2	Sample batch exit module	A-4
A.3	Sample CICS exit module	A-8

A.1 DATBASBF and DATBASAF exits

The CA-IDMS/Total Transparency supports two user exits:

- **DATBASBF** — intercepts parameter lists passed from the application and inspects commands being issued
- **DATBASAF** — intercepts parameter lists returned to the application and inspects commands being issued

DATBASBF and DATBASAF exits are invoked if they are linked with the ETOTBINT module that is linked with the application.

The transparency supports Release 8 of Total. Releases 5, 6, and 7 of TOTAL are supported through the user exits ETOT07EX and ETOT56EX.

DATBASBF: DATBASBF allows the inspection of incoming parameters and subsequent auditing as necessary. Linkage conventions for DATBASBF are defined below:

R1 = Incoming parameter list
R2 = Address of subschema control block
R13 = Save area for exit use
R14 = Return address
R15 = Return code
 F'0' = Continue processing
 F'4' = Bypass processing, return to application

DATBASAF: DATBASAF is invoked after database processing and before control is returned to the application program. Linkage conventions for DATBASAF are defined below:

R1 = Return parameter list (same as the incoming parameter list)
R2 = Address of subschema control block
R13 = Save area for exit use
R14 = Return address
R15 = Return code
 F'0' = Continue processing;
 nonzero return code will invoke an abend

A.2 Sample batch exit module

The following DATBASBF module can be used as a guide to coding transparency exits for batch applications.

```

        TITLE 'DATBASBF - SAMPLE BATCH TRANSPARENCY EXIT '
DATBASBF CSECT
        USING *,R15
        USING SSC,R2                SSC ADDRESSABILITY
        SPACE 2

*
*   REGISTER EQUATES
*
R0      EQU    0
R1      EQU    1
R2      EQU    2
R3      EQU    3
R4      EQU    4
R5      EQU    5
R6      EQU    6
R7      EQU    7
R8      EQU    8
R9      EQU    9
R10     EQU    10
R11     EQU    11
R12     EQU    12
R13     EQU    13
R14     EQU    14
R15     EQU    15
        EJECT
        @MODE MODE=BATCH,QUOTES=YES,DEBUG=YES
        SPACE 2

*-----*
*
*   DATBASBF - SAMPLE BATCH CA-IDMS TOTAL/T EXIT MODULE
*
*   THIS IS AN EXAMPLE OF A BATCH BEFORE CALL EXIT MODULE.
*   IT'S PURPOSE IS TO BE A GUIDE TO CODING TOTAL/T EXITS.
*   THIS SAMPLE ILLUSTRATES:
*       HOW TO EXAMINE APPL PARMS FOR CALL TYPE.
*       HOW TO SET UP SUBSCHEMA CONTROL FOR DML CALLS.
*       HOW TO CALL IDMS WITH RETURN TO THE EXIT.
*   ALSO INCLUDED:
*       ACQUISITION OF ADDITIONAL WORKAREA STORAGE.
*
*   WHEN LINKED TO ETOTBINT, CONTROL WILL BE PASSED TO
*   THIS MODULE FOR EACH APPLICATION CALL TO TOTAL/T,
*   BEFORE THE TOTAL/T PROCESSING OF THE APPL CALL.
*
*   AT ENTRY:
*
*   R1 - A(APPLICATION PARM LIST)
*   R2 - A(APPLICATION SUBSCHEMA CONTROL)  * SEE NOTE(S) *
*   R13 - A(SAVEAREA)
*   R14 - RETURN ADDR (WITHIN ETOTBINT)
*   R15 - A(DATBASBF)
*

```

```

*      NOTE: THE SUBSCHEMA CONTROL IDMS/DC EXTENSION IS USED BY      *
*      THIS EXIT AS A REENTRANT AREA FOR PROGRAM STORAGE.             *
*                                                                      *
*      NOTE: NO IDMS DML SHOULD BE ATTEMPTED DURING A SINON CALL.    *
*                                                                      *
*      NOTE: THE ONLY DIFFERENCE BETWEEN THIS BEFORE CALL EXIT      *
*      AND AN AFTER CALL EXIT (DATBASAF) IS THAT CONTROL WILL BE    *
*      PASSED TO DATBASAF AFTER TOTAL/T PROCESSING OF THE CALL.      *
*                                                                      *
*-----*
      SPACE 1
*
*      SAVE REGISTERS
*
      STM  R14,R12,12(R13)      SAVE INCOMING REGISTERS
      ST   R13,EXITR13          SAVE SAVEAREA REGISTER
      LR   R10,R15              SET  BASE
      DROP R15
      USING DATBASBF,R10        EXIT ADDRESSABILITY
      TITLE 'DATBASBF - PROCESS CALL TYPE '
*
*
*      PROCESS CALL TYPE.
*
*
      L     R3,0(,R1)           --> PARM #1
*
*      SINON CALL ??
*
      CLC   0(5,R3),=CL5'SINON' SINON CALL ??
      BE    SINON000            YES, BR
*
*      SINOF CALL ??
*
      CLC   0(5,R3),=CL5'SINOF' SINOF CALL ??
      BE    SINOF000            YES
      B     EXITRTN0            ELSE, EXIT
      TITLE 'DATBASBF - SINON PROCESSING'
*
*
*      PROCESS SINON CALL
*
*
SINON000 LA  R14,EXITSTGL        LENGTH OF
          ST  R14,EXITSTGA      NEEDED STORAGE
*
          #GETSTG TYPE=(USER, LONG), LEN=(R14), ADDR=(R1),          X
          INIT=0, LOC=BELOW
*
          LTR  R15,R15          STORAGE ACQUIRED ??
          BNZ  EXITRTN8         NO, REQUEST ABEND
          ST   R1,EXITSTGA      SAVE STORAGE ADDR
          B    EXITRTN0         EXIT EXIT
          TITLE 'DATBASBF - SINOF PROCESSING'
*
*
*      PROCESS SINOF CALL.
*

```

```
*
SINOF000 BAL   R5,ROLLBAK0          GO ROLLBAK
          B     EXITRTN0             EXIT
          TITLE 'DATBASBF - ISSUE COMMIT '

*
*
*       ISSUE COMMIT.
*
*
COMMIT00 @COMMIT CALL=NO              SETUP
          BAL   R8,CALLIDMS           ISSUE COMMIT
          BR    R5
          TITLE 'DATBASBF - ISSUE ROLLBAK '

*
*
*       ISSUE ROLLBAK.
*
*
ROLLBAK0 @ROLLBAK CONTINUE,CALL=NO    SETUP
          BAL   R8,CALLIDMS           ISSUE ROLLBAK
          BR    R5
          TITLE 'DATBASBF - CALL IDMS '

*
*
*       CALL IDMS.
*
*
CALLIDMS LA    R1,SSCTPARM            --> IDMS PARMLIST
          LA    R13,EXITSAVE          A(SAVEAREA)
          L     R15,=V(IDMS)          A(IDMS)
          BALR  R14,R15               CALL IDMS
          BR    R8                   RETURN
          TITLE 'DATBASBF - RETURN PROCESSING '

*
*
*       RETURN RC=0 (CONTINUE OK)
*
*
EXITRTN0 L     R13,EXITR13             RESTORE R13
          LM    R14,R12,12(R13)        RESTORE REGS
          XR    R15,R15               SET RC=0
          BR    R14                   RETURN TO ETOTBINT

*
*
*       RETURN RC=4 (SKIP CURRENT APPLICATION CALL)
*
*
EXITRTN4 L     R13,EXITR13             RESTORE R13
          LM    R14,R12,12(R13)        RESTORE REGS
          LA    R15,4                 SET RC=4
          BR    R14                   RETURN TO ETOTBINT

*
*
*       RETURN RC=8 (ABEND REQUEST)
*
*
EXITRTN8 L     R13,EXITR13             RESTORE R13
          LM    R14,R12,12(R13)        RESTORE REGS
          LA    R15,8                 SET RC=8
```

```
BR      R14                      RETURN TO ETOTBINT
TITLE 'DATBASBF - DSECTS '
COPY   #SSC120
ORG     DCCDCBMS

*
*      EXIT WORKAREA (IN SUBSCHEMA CONTROL DC EXTENSION).
*
EXITPRMS EQU *
ADDR1   DS   A                   SPARE
ADDR2   DS   A                   SPARE
ADDR3   DS   A                   SPARE
*
DMLSEQ  DS   F                   DUMMY DMLSEQ
        ORG   DMLSEQ
*
SYSPLIST EQU *                   EXIT SAVEAREA(S)
EXITSAVE DS  18F                 PLIST AREA
EXITR13  DS   F                 IDMS CALL SAVE
*
EXITSTG  EQU *                   EXIT STORAGE AREA
EXITSTGL EQU 1024                L'SORAGE (ADJUST AS NEEDED)
EXITSTGA DS   F                 A(EXIT STORAGE)
        ORG
DATBASBF CSECT
        END
```

A.3 Sample CICS exit module

The following DATBASBF module can be used as a guide to coding transparency exits for CICS applications.

```

        TITLE 'DATBASBF - SAMPLE CICS TRANSPARENCY EXIT '
DATBASBF CSECT
        USING *,R15
        USING SSC,R2          SSC ADDRESSABILITY
        USING DFHEISTG,R12     EIS ADDRESSABILITY
DFHEIPLR EQU 12
        SPACE 2

*
*   REGISTER EQUATES
*
R0      EQU 0
R1      EQU 1
R2      EQU 2
R3      EQU 3
R4      EQU 4
R5      EQU 5
R6      EQU 6
R7      EQU 7
R8      EQU 8
R9      EQU 9
R10     EQU 10
R11     EQU 11
R12     EQU 12
R13     EQU 13
R14     EQU 14
R15     EQU 15
        EJECT
        @MODE MODE=CICS,QUOTES=YES,DEBUG=YES
        SPACE 2

*-----*
*
*   DATBASBF - SAMPLE CICS CA-IDMS TOTAL/T EXIT MODULE
*
*   THIS IS AN EXAMPLE OF A CICS BEFORE CALL EXIT MODULE.
*   IT'S PURPOSE IS TO BE A GUIDE TO CODING TOTAL/T EXITS.
*   THIS SAMPLE ILLUSTRATES:
*       HOW TO EXAMINE APPL PARMS FOR CALL TYPE.
*       HOW TO SET UP SUBSCHEMA CONTROL FOR DML CALLS.
*       HOW TO CALL IDMS WITH RETURN TO THE EXIT.
*   ALSO INCLUDED:
*       ACQUISITION OF ADDITIONAL WORKAREA STORAGE.
*
*   WHEN LINKED TO IDMSINTC, CONTROL WILL BE PASSED TO
*   THIS MODULE FOR EACH APPLICATION CALL TO TOTAL/T,
*   PRIOR TO ANY TOTAL/T PROCESSING OF THE APPL CALL.
*
*   AT ENTRY:
*
*   R1 - A(APPLICATION PARM LIST)
*   R2 - A(APPLICATION SUBSCHEMA CONTROL)  * SEE NOTE(S) *
*   R12 - A(CICS EXEC INTERFACE STORAGE)
*   R13 - A(REENTRANT SAVEAREA)

```



```

*      R14 - RETURN ADDR (WITHIN IDMSINTC)      *
*      R15 - A(DATBASBF)                        *
*
*      NOTE: THIS EXIT MUST BE LINKED TO IDMSINTC.  *
*
*      NOTE: COMMAND LEVEL CALLS SHOULD BE USED TO ACCESS CICS  *
*      SERVICES.  EVEN IF THE APPLICATION IS MACRO LEVEL, A    *
*      COMMAND LEVEL ENVIRONMENT IS ACTIVE WHEN THIS EXIT GETS  *
*      CONTROL.  USE OF MACRO LEVEL SERVICES IS NOT RECOMMENDED. *
*      COMMAND LEVEL CALLS REQUIRE USE OF A CICS PREPROCESSOR.  *
*
*      NOTE: NO IDMS DML SHOULD BE ATTEMPTED DURING A SINON CALL. *
*
*      NOTE: THE ONLY DIFFERENCE BETWEEN THIS BEFORE CALL EXIT  *
*      AND AN AFTER CALL EXIT (DATBASAF) IS THAT CONTROL WILL BE *
*      PASSED TO DATBASAF AFTER TOTAL/T PROCESSING OF THE CALL.  *
*
*-----*
*      SPACE 1
*
*      SAVE REGISTERS
*
*      STM  R14,R12,12(R13)      SAVE INCOMING REGISTERS
*      ST   R13,EXITR13          SAVE SAVEAREA REGISTER
*      LR   R10,R15              SET BASE
*      DROP R15
*      USING DATBASBF,R10        EXIT ADDRESSABILITY
*      TITLE 'DATBASBF - PROCESS CALL TYPE '
*
*
*      PROCESS CALL TYPE.
*
*
*      L     R3,0(,R1)           --> PARM #1
*
*      SINON CALL ??
*
*      CLC   0(5,R3),=CL5'SINON' SINON CALL ??
*      BE    SINON000            YES, BR
*
*      SINOF CALL ??
*
*      CLC   0(5,R3),=CL5'SINOF' SINOF CALL ??
*      BE    SINOF000            YES
*      B     EXITRTN0            ELSE, EXIT
*      TITLE 'DATBASBF - SINON PROCESSING'
*
*
*      PROCESS SINON CALL
*
*      SINON000 LA  R14,EXITSTGL      LENGTH OF
*                STH R14,EXITSTGA      NEEDED STORAGE
*
*      EXEC CICS GETMAIN SET(R15)      X
*                INITIMG(X'00')      X
*                FLENGTH(EXITSTGA) .
*
*      ST     R15,EXITSTGA            SAVE STORAGE ADDR

```

```

      B      EXITRTN0          EXIT EXIT
      TITLE 'DATBASBF - SINOF PROCESSING'
*
*
*      PROCESS SINOF CALL.
*
*
SINOF000 BAL   R5,ROLLBAK0          GO ROLLBAK
          B     EXITRTN0          EXIT
          TITLE 'DATBASBF - ISSUE COMMIT '
*
*
*      ISSUE COMMIT.
*
*
COMMIT00 @COMMIT CALL=NO          SETUP
          BAL   R8,CALLIDMS        ISSUE COMMIT
          BR    R5
          TITLE 'DATBASBF - ISSUE ROLLBAK '
*
*
*      ISSUE ROLLBAK.
*
*
ROLLBAK0 @ROLLBAK CONTINUE,CALL=NO  SETUP
          BAL   R8,CALLIDMS        ISSUE ROLLBAK
          BR    R5
          TITLE 'DATBASBF - CALL IDMS '
*
*
*      CALL IDMS.
*
*
CALLIDMS LA    R1,SSCTPARM          --> IDMS PARMLIST
          LA    R13,EXITSAVE        A(SAVEAREA)
          L     R15,ADDRINC1        A(IDMSINC1)
          BALR  R14,R15             CALL IDMS
          DC    CL8'$ESCPT$'        CALL TAG (REQUIRED)
          BR    R8                  RETURN
          TITLE 'DATBASBF - RETURN PROCESSING '
*
*
*      RETURN RC=0 (CONTINUE OK)
*
*
EXITRTN0 L     R13,EXITR13          RESTORE R13
          LM    R14,R12,12(R13)     RESTORE REGS
          XR    R15,R15             SET RC=0
          BR    R14                 RETURN TO IDMSINTC
*
*
*      RETURN RC=4 (SKIP CURRENT APPLICATION CALL)
*
*
EXITRTN4 L     R13,EXITR13          RESTORE R13
          LM    R14,R12,12(R13)     RESTORE REGS
          LA    R15,4               SET RC=4
          BR    R14                 RETURN TO IDMSINTC
*

```

```

*
*      RETURN RC=8 (ABEND REQUEST)
*
*
EXITRTN8 L      R13,EXITR13          RESTORE R13
          LM      R14,R12,12(R13)    RESTORE REGS
          LA      R15,8              SET RC=8
          BR      R14                RETURN TO IDMSINTC
          TITLE   'DATBASBF - DSECTS '
          COPY    #SSC120
          ORG     DCCDCBMS

*
*      EXIT WORKAREA (IN SUBSCHEMA CONTROL DC EXTENSION).
*
EXITPRMS EQU     *
          DS      A                  SPARE
ADDRINC1 DS      A                  A(IDMSINC1)
          DS      A                  SPARE

*
DMLSEQ   DS      F                  DUMMY DMLSEQ
          ORG     DMLSEQ

*
EXITSAVE DS      18F                EXIT SAVEAREA(S)
          DS      F                  IDMS CALL SAVE
EXITR13  DS      F                  EXIT R13  SAVE

*
EXITSTG  EQU     *                  EXIT STORAGE AREA
EXITSTGL EQU     1024                L'SORAGE (ADJUST AS NEEDED)
EXITSTGA DS      F                  A(EXIT STORAGE)
          ORG
          DFHEISTG ,
DATBASBF CSECT
          END

```


Appendix B. Generator Messages

B.1 About generator messages	B-3
B.2 Error messages	B-4
B.3 Warning message	B-7
B.4 Fatal messages	B-8

B.1 About generator messages

When the generator detects a processing error, the generator writes a message to an error report. On the report, the message follows the record that contains the error. An asterisk is printed under the first character of the entry causing the error or warning. An E, W, or F labels each message, as follows:

Label	Status	Effect on processing
E	Nonfatal error	Generator continues processing
W	Warning	Generator continues processing
F	Fatal error	Generator terminates processing

This appendix lists and describes error, warning, and fatal messages issued by the generator.

B.2 Error messages

Message issued	Reason
BAD GROUP ELEMENT LENGTH	Error in Total DDL; the specified group-length must match the combined lengths of the individual items making up the group.
CARD BUFFER FULL (ETOTPARS)	More than 500 SYNONYM, AREA, SCHEMA, DEVICE-MEDIA, SUBSCHEMA, and USAGE-MODE statements; relink ETOTMAIN to include ETOTR002 (see Appendix E, “Increasing Generator Input Limits” on page E-1).
IMPROPER ELEMENT NAME	Invalid Total DDL data name or keyword; entry must be *FILLER*, RECORD-CODE, or an 8-character data name.
IMPROPER FILE NAME	Invalid file name; entry must be four characters in length and alphanumeric.
IMPROPER INTEGER	Expecting an integer; entry is nonnumeric.
IMPROPER SYNONYM	Invalid or missing CA-IDMS/DB element, record, or set name in SYNONYMS control statement; entry must be alphanumeric.
IMPROPER SYNONYM SOURCE	Invalid Total element or file name or transparency set name in SYNONYMS control statement; entry must be 4, 8, or 12 characters long.
IMPROPER SYNTAX	Invalid Total DDL syntax.
INVALID AREA NAME	Invalid or missing CA-IDMS/DB area name in AREA or USAGE-MODE control statement; entry must be from 1 to 16 characters long and must be alphanumeric.

Message issued	Reason
INVALID LOAD LRECL	Invalid logical record length in LRECL entry of LOAD control statement; entry must be greater than or equal to 1.
INVALID LOAD SYNTAX	Invalid LOAD statement syntax; entry following keyword LOAD must be ALL, ALL., or LRECL.
INVALID SUBSCHEMA NAME	Invalid or missing CA-IDMS/DB subschema name in SUBSCHEMA NAME IS entry of SUBSCHEMA control statement; entry must be from 1 to 8 characters long.
INVALID TASK NAME	Invalid Total task name in USAGE-MODE control statement; entry must be 8 characters long and must be alphanumeric.
INVALID USAGE	Invalid CA-IDMS/DB usage mode in USAGE-MODE control statement; entry must specify RETRIEVAL, UPDATE, SHARED RETRIEVAL, PROTECTED RETRIEVAL, EXCLUSIVE RETRIEVAL, SHARED UPDATE, PROTECTED UPDATE, or EXCLUSIVE UPDATE.
INVALID USE OF READY	A READY ALL USAGE-MODE control statement was detected after a READY <i>area-name</i> USAGE-MODE control statement had been found, or vice versa.
INVALID USE OF PGMNAME	A <i>total-task-name</i> USAGE-MODE control statement specifying an <i>idms-area-name</i> was detected after a <i>total-task-name</i> ALL statement had been found, or vice versa.
MISSING BEGIN-DATA-BASE-GENERATION	The beginning statement of the Total DDL file is missing.

Message issued	Reason
SYMBOL-TABLE FULL	The number of files plus the number of elements in the entire DDL is greater than the table size allowed in the transparency.
UNDEFINED AREA	Unrecognized CA-IDMS/DB area name in USAGE-MODE control statement; entry must be defined in SCHEMA control statement.
UNDEFINED ELEMENT	Unrecognized Total data-element name; entry must be defined in Total DDL.
UNDEFINED FILE	Unrecognized Total file name in Total DDL, SYNONYMS control statement, or AREA control statement. Any Total file name used as the first four characters of a Total file element name in Total DDL must also be defined in Total DDL; a Total file name used to qualify a Total element name in the SYNONYMS control statement must be defined in the Total DDL; a Total file name entered in the AREA control statement must be defined in the Total DDL.
UNDEFINED SET NAME	Unrecognized transparency set name in SYNONYMS control statement; entry must be the name of a CA-IDMS/DB set established by the generator.

B.3 Warning message

Message issued	Reason
NEAREST PRIME TOO LARGE FOR HASH TABLE	Insufficient table size.

B.4 Fatal messages

Message issued	Reason
REACHED EOF ON SKEL-FILE	Invalid or missing loader skeleton (ETOTLOAD); probable JCL error.
REACHED EOF ON DB-TABLES	Invalid ETOTTBL input; probable JCL error.

Appendix C. Loader Messages

C.1 Loader messages issued	C-3
--------------------------------------	-----

C.1 Loader messages issued

The following messages are issued by the CA-IDMS/Total Transparency loader.

Message issued	Reason
??? FILE NOT FOUND: INPUT: <i>input-record-name</i>	The unloaded Total database includes a record (<i>input-record-name</i>) that references a Total file not known to the loader. Ensure that the loader is reading the proper Total data and that the Total database was unloaded correctly.
??? STRANGE FILE TYPE	The array built into the WORKING-STORAGE SECTION of the transparency loader contains an invalid file-type field.
??? NOT ENOUGH OWNER DESCRIPTOR RECORDS	A transparency buffer limit has been exceeded.

Appendix D. Runtime Messages

D.1	Three classes of error codes	D-3
D.2	Total DML syntax error codes	D-4
D.3	Error conditions detected by CA-IDMS/DB	D-5
D.4	Transparency errors	D-8

D.1 Three classes of error codes

When you process database requests from a Total application program, the transparency runtime interface returns three classes of error codes to the program, as follows:

- **Syntax errors** — codes resulting from syntax errors detected before CA-IDMS/DB is called to perform the requested database service. These codes are in the form of standard Total error-status codes and reflect errors found in Total DML syntax.
- **Errors detected by CA-IDMS/DB** — codes resulting from errors or possible error conditions detected by CA-IDMS/DB while attempting to perform the requested database service. These codes are in the form of Total equivalents to the status codes returned to the transparency runtime interface by CA-IDMS/DB.
- **Transparency errors** — codes resulting from internal transparency conditions. The runtime interface returns either a CA-IDMS/DB status code or a transparency code that resembles a standard Total error-status code.

This appendix describes these three classes of codes.

D.2 Total DML syntax error codes

When the transparency runtime interface receives a request from a Total application program, it checks for DML syntax errors by performing a Total error-checking routine before calling CA-IDMS/DB. The runtime interface returns the results of this error-checking routine in the form of standard Total error-status codes. Refer to the appropriate Total documentation for an explanation of these codes.

D.3 Error conditions detected by CA-IDMS/DB

When the transparency runtime interface issues a call to CA-IDMS/DB to perform a database service requested by the Total application program, CA-IDMS/DB processes (or attempts to process) the request and returns a CA-IDMS/DB status code to the runtime interface. The runtime interface translates this status code to an equivalent Total error-status code and returns the Total code to the requesting Total application.

Corresponding Total and CA-IDMS/DB status codes: The following table lists Total status codes and the corresponding CA-IDMS/DB status codes. Some Total status codes are issued due to situations other than the return of a CA-IDMS/DB status code resulting from a CA-IDMS DML call. In cases where there is no reasonable Total status code to match a CA-IDMS/DB status code, the actual CA-IDMS/DB status code is used as the Total status. A CA-IDMS/DB status code used instead of a Total status indicates an unusual situation.

If a status code is returned that is not noted in this table, refer to the appropriate Total error codes manual or to *CA-IDMS Messages and Codes*.

Total status	CA-IDMS/DB status	Description
****	0000	OK status
CICS	1468	CICS interface not started
IDMS	0000	IDMS STATISTICS CALL return code
FNOP	0301	Either area not readied or ready failed
	1201	Either area not readied or ready failed
	0208	Invalid record or set name
	0308	Invalid record or set name
	0508	Invalid record or set name
	0708	Invalid record or set name
	1108	Invalid record or set name
	1208	Invalid record or set name
	0210	Function not allowed
	0310	Function not allowed
	0510	Function not allowed
	0710	Function not allowed
	0810	Function not allowed
	0910	Function not allowed

Total status	CA-IDMS/DB status	Description
	1110	Function not allowed
	1210	Function not allowed
	0323	Invalid area or parameter list
IRLC	0302	DBKEY out of page range
	1202	DBKEY out of page range
	0371	Page range not found in DMCL
	0971	Page range not found in DMCL
DUPM	0705	Duplicates not allowed
	0805	Duplicates not allowed
	1205	Duplicates not allowed
NHLD	0306	Currency not established
	0706	Currency not established
	0806	Currency not established
	1106	Currency not established
	1506	Currency not established
	1606	Currency not established
	0313	No current of run unit
	0513	No current of run unit
	0813	No current of run unit
	1613	Currency not established
	0220	Current record not the same type as the named record
	0520	Current record not the same type as the named record
	0820	Current record not the same type as the named record
	0225	Currency not established
	0825	Currency not established
	1225	Currency not established
LOCK	0209	Named area readied in incorrect usage mode
	0709	Named area readied in incorrect usage mode

Total status	CA-IDMS/DB status	Description
	0809	Named area readied in incorrect usage mode
	1109	Named area readied in incorrect usage mode
	1209	Named area readied in incorrect usage mode
	0221	Area other than the named area was readied in the incorrect usage mode
	0721	Area other than the named area was readied in the incorrect usage mode
	0821	Area other than the named area was readied in the incorrect usage mode
	1121	Area other than the named area was readied in the incorrect usage mode
	1221	Area other than the named area was readied in the incorrect usage mode
	0966	Area not available for update
FULL	1211	Area full
NOIO	0318	Record not bound
	0518	Record not bound
	1218	Record not bound
RSRV	0928	Named areas previously readied
IMDL	0230	Record is owner of a nonempty set
FTYP	0331	Storage mode conflict
IOER	0970	Database or journal file will not open properly
NACT	1400	Run unit not bound
	1469	Run unit not bound
FNAV	1408	Invalid record or set name
	1508	Invalid record or set name
ICOR	1472	Insufficient for LOAD or STORE
TFUL	1473	Run unit maximum reached
ACTV	1477	Run unit bound twice

D.4 Transparency errors

When the transparency runtime interface receives a Total application program request that it cannot process because all the transparency requirements for handling the request are not met, or when the runtime interface receives a status code from CA-IDMS/DB for which there is no Total equivalent, the runtime interface issues one of the following codes:

- *IDMS status code*
- CICS
- USxx

IDMS status code: The runtime interface receives an CA-IDMS/DB status code in response to a request for database services; a Total equivalent to this request does not exist. For an explanation of CA-IDMS/DB status codes, refer to *CA-IDMS Messages and Codes*.

CICS: The CICS code indicates that the interface between the central version and CICS has not been started. The transparency runtime interface has issued a call to CA-IDMS/DB to process a request from a Total CICS application, and CA-IDMS/DB has returned a status code of *nn68*.

Note: Minor code 68 can be issued with any major code.

US: A US status code indicates that the transparency runtime interface cannot process a DML command issued by the Total application. US status codes can result from any of the following errors in a Total DML command:

- The key needed to perform the CA-IDMS/DB equivalent of a Total store command for a variable file is not provided in the data-list parameter
- A Total store command has been issued for a coded variable file, but the CA-IDMS/DB equivalent of the record code in the data area is not defined in the subschema
- The linkpath named in a Total linkpath parameter is not valid for the record code in the data area
- The data-list parameter in the application call to ETOTBINT or ETOTCINT is not correctly punctuated
- ETOTTBL was not found. During SINON processing, the runtime control table (ETOTTBL), designated by the DBMOD parameter or the SINON call, could not be loaded or located.
- ETOTTRAN was not found (batch only). If ETOTTRAN was not linked during the link edit of ETOTBINT, a copy of ETOTTRAN could not be loaded or located.
- Runtime work area storage was not available. At the time the application issued a SINON call, enough runtime work area storage for processing was not available.

- A user exit has requested an abend
- RQLOC function was invoked under central version. This function is available in local mode only.
- RQLOC-FILE (record name) was not found in subschema
- RQLOC-FILE (record name) was not found in ETOTTBL

US status codes: The following table lists US status codes.

Total status	Meaning
US01	Internal hashing error — REGEN ETOTTBL
US02	Record buffer too small
US03	Set key missing
US04	Missing set relation
US05	Improper key connect
US06	Invalid record code
US07	Insufficient storage (batch)
US08	Error in load of ETOTTRAN or ETOTTBL
US09	RQLOC function invoked under central version
US10	RQLOC record or file name not found in the subschema
US99	Element table not correct for **REST**; table must be regenerated
USEX	Abend request from user exit

Appendix E. Increasing Generator Input Limits

E.1	ETOTMAIN input limits	E-3
E.1.1	Limits on Total DDL statements	E-3
E.1.2	Limits on transparency generator control statements	E-3

E.1 ETOTMAIN input limits

You may need to increase transparency generator input limits to accommodate large database conversions. The transparency generator (ETOTMAIN) comes in two versions. These versions differ from each other in the capacity for input to the generator.

Limits apply to:

- Total DDL statements
- Transparency generator control statements

The default installation link-edit of the transparency generator (ETOTMAIN) includes ETOTROOT, which sets lower limits on input to the transparency generator. These limits accommodate most uses of the transparency.

If you are working with larger databases and require larger limits, relink ETOTMAIN with ETOTROO2.

Note that you should *not* relink ETOTMAIN unless necessary. ETOTROO2 will increase the memory requirements of ETOTMAIN. The following tables show the default limits and optional larger limits for the two types of input to the transparency generator.

E.1.1 Limits on Total DDL statements

Item	Default limit	ETOTROO2 limit
Files	300	1000
Elements	5900	9000
Linkpaths (=sets)	500	500
Record codes	500	500

E.1.2 Limits on transparency generator control statements

Item	Default limit	ETOTROO2 limit
16-byte synonyms (files, records, sets)	2000	2000
32-byte synonyms (elements)	1000	1000
Usage mode statements	2000	2000
Schema/subschema statements	500	5000

Appendix F. OS/390 Job Control Language

F.1	Generate (ETOTMAIN)	F-3
F.2	Assembling and link editing ETOTTBL	F-5
F.3	Compiling and link editing ETOTLOAD	F-6
F.4	Loading the database with ETOTLOAD	F-8
F.5	Assembling the CICS transparency interface (ETOTCINT)	F-10
F.6	Creating the CICS CA-IDMS/DB Interface (IDMSINTC)	F-11
F.7	Batch application program link edit	F-12
F.8	CICS application program link edit	F-13
F.9	Runtime JCL	F-14

F.1 Generate (ETOTMAIN)

Use the following JCL to execute the transparency generator (ETOTMAIN):

ETOTMAIN (OS/390)

```
//GENERATE EXEC PGM=ETOTMAIN,REGION=2048K
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
// DD DSN=idms.loadlib,DISP=SHR
//SYSLST DD SYSOUT=A
//dcmsg DD DSN=idms.sysmsg.ddl dcmsg,DISP=SHR
//SYSIDMS DD *
DMCL=etotdmc1
Input other SYSIDMS parameters, as required
/*
//SYSIPT DD DSN=user.srclib(totalddl),DISP=SHR
// DD DSN=user.srclib(ctrlstmt),DISP=SHR
// DD DSN=idms.srclib(ETOTTBL),DISP=SHR
// DD DSN=idms.srclib(ETOTLOAD),DISP=SHR
//SYSPCH DD DSN=user.etotschm,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH03 DD DSN=user.etotsubs,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH04 DD DSN=user.etottbl,UNIT=disk,DISP=(NEW,CATLG,
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
//SYSPCH05 DD DSN=user.etotload,UNIT=disk,DISP=(NEW,CATL),
SPACE=(TRK,(4,4)),DCB=(RECFM=FB,LRECL=80,BLKSIZE=bbbb)
```

Note: The punched output should be directed to sequential files. Do not use partitioned datasets.

idms.dba.loadlib	Dataset name of the CA-IDMS/DB load library containing the DMCL and database name table load modules
idms.loadlib	Dataset name of the CA-IDMS/DB load library containing CA-IDMS executable modules
dcmsg	DDname of the system message area
idms.sysmsg.ddl dcmsg	Dataset name of the system message area
etotdmc1	Name of the DMCL to be accessed at runtime; use the name of the global DMCL ►► See the <i>CA-IDMS Database Administration</i> , for more information about SYSIDMS parameters.
user.srclib	Dataset name of the user source library
totalddl	Name of the library member containing Total DDL
idms.srclib	Dataset name of the CA-IDMS/DB source library
ctrlstmt	Name of the library member containing user-supplied control statements

user.etotschm	Dataset name of the file containing the schema source generated by ETOTMAIN. Run this output through the schema compiler (IDMSCHEM).
user.etotsubs	Dataset name of the file containing the subschema source generated by ETOTMAIN. Run this output through the subschema compiler (IDMSUBSC).
user.etottbl	Dataset name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL. Run this output through the assembly and link edit process.
user.etotload	Dataset name of the file containing the customized loader generated by ETOTMAIN from the skeleton loader, ETOTLOAD. Run this output through the COBOL compile and link edit process.
disk	Symbolic device name of the file containing the punched output
bbbb	Block size of the file containing the punched output; must be a multiple of 80

F.2 Assembling and link editing ETOTTBL

Use the JCL below to assemble and link edit ETOTTBL.

ETOTTBL (OS/390)

```
//ASM      EXEC PGM=IEUASM,PARM='NOLOAD,DECK'
//SYSPRINT DD   SYSOUT=A
//SYSLIB   DD   DSN=idms.maclib,DISP=SHR
//         DD   DSN=sys1.maclib,DISP=SHR
//         DD   DSN=cics.maclib,DISP=SHR
//SYSUT1   DD   UNIT=disk,SPACE=(CYL,(2,2))
//SYSUT2   DD   UNIT=disk,SPACE=(CYL,(2,2))
//SYSUT3   DD   UNIT=disk,SPACE=(CYL,(2,2))
//SYSPUNCH DD   DSN=&&ETOTTBL,UNIT=disk,SPACE=(80,(200,50)),
//           DISP=(NEW,PASS)
//SYSIN    DD   DSN=user.etottbl,DISP=SHR
/*
//LINKEDIT EXEC PGM=IEWL,PARM=(XREF,LET,LIST),REGION=256K
//SYSPRINT DD   SYSOUT=A
//SYSUT1   DD   UNIT=disk,SPACE=(CYL,(2,2))
//SYSLMOD  DD   DSN=idms.loadlib,DISP=SHR
//SYSLIN   DD   DSN=&&ETOTTBL,DISP=(OLD,DELETE)
//         DD   DDname=SYSIN
//SYSIN    DD   *
          NAME etottbl(R)
/*
```

idms.maclib	Dataset name of the CA-IDMS/DB macro library
cics.maclib	Dataset name of the CICS macro library
disk	Symbolic device name of the disk file
etottbl	Name of the customized transparency ETOTTBL load module, which replaces the Total DBMOD module
sys1.maclib	Dataset name of the system macro library
user.etottbl	Dataset name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL
idms.loadlib	Dataset name of the CA-IDMS/DB load library

F.3 Compiling and link editing ETOTLOAD

Use the following JCL to compile and link edit ETOTLOAD.

ETOTLOAD (OS/390)

```

//*****
//* COBOL compile
//*****
//COMP      EXEC PGM=IKFCBL00,REGION=256K,
//          PARM='DECK,NOLOAD,NOLIB,BUF=50000,SIZE=150K,DMAP,PMAP'
//SYSPRINT DD SYSOUT=*
//SYSLIB    DD DSN=sys1.coblib,DISP=SHR
//SYSUT1    DD UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT2    DD UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT3    DD UNIT=disk,SPACE=(TRK,(10,5))
//SYSUT4    DD UNIT=disk,SPACE=(TRK,(10,5))
//SYSPUNCH DD DSN=&b,DISP=(NEW,PASS),UNIT=disk,
//          SPACE=(80,(400,40))
//SYSIN     DD DSN=user.etotload,DISP=SHR
//*****
//* Link edit the object module created in the COBOL compile
//*****
//LINK      EXEC PGM=IEWL,PARM='LIST,XREF,LET,SIZE=512K,96K)'
//SYSPRINT DD SYSOUT=*
//SYSLIB    DD DSN=sys1.coblib,DISP=SHR
//SYSUT1    DD UNIT=disk,SPACE=(TRK,(20,5))
//LIB       DD DSN=idms.loadlib,DISP=SHR
//SYSLMOD   DD DSN=idms.loadlib,DISP=SHR
//SYSLIN    DD DSN=&b,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSIN     DD *
MODE RMODE(24),AMODE(24)
INCLUDE LIB(IDMSDBLU)
INCLUDE LIB(IDMS)
INCLUDE LIB(etotsubs)
ENTRY etotload
NAME etotload(R)

```

Specifying an object library in the LINK step: In the LIB DD statement, you can specify a CA-IDMS/DB object library rather than a load library. If you do this you must include the following statement as part of SYSIN for the LINK step:

```
INCLUDE LIB(IDMSCALC,IDMSUTIL,IDMSALIO)
```

sys1.coblib	Dataset name of file containing COBOL support modules
disk	Symbolic device name
& b	Punch file containing object code from the COBOL compile step
user.etotload	Dataset name of the file containing the customized loader generated by ETOTMAIN
idms.loadlib	Dataset name of the CA-IDMS/DB library containing CA-IDMS executable modules

etotsubs	Name of the subschema, as specified in the transparency generator control statements
etotload	Name of the customized load program

F.4 Loading the database with ETOTLOAD

Use the following JCL to execute ETOTLOAD. Output from this step is processed by the FASTLOAD utility.

►►Refer to *CA-IDMS Utilities* for more information about the FASTLOAD utility statement.

ETOTLOAD (OS/390)

```
//*****
//* Use ETOTLOAD to create the load file for FASTLOAD
//* (unloaded Total database is converted to CA-IDMS/DB format)
//*****
//DBL1      EXEC PGM=ETOTLOAD,REGION=512K
//STEPLIB   DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//unload    DD DSN=total.data,DISP=SHR
//SYSIDMS   DD *
//          DMCL=etotdmcl DBNAME=etotal
/*
//dcmmsg     DD DSN=idms.sysmsg.ddldcmmsg,DISP=SHR
//SYSUDUMP   DD SYSOUT=*
//SYSOUD     DD SYSOUT=*
//SYSOUT     DD SYSOUT=*
//SYSLST     DD SYSOUT=*
//SYSJRNL    DD DUMMY
//RELDCTL    DD DSN=reldctl,DISP=(NEW,CATLG),UNIT=disk,
//          VOL=SER=nnnnnn,
//          DCB=(RECFM=FB,LRECL=60,BLKSIZE=1200),
//          SPACE=(TRK,(10,10),RLSE)
//SYS002     DD DSN=sortdbl1,DISP=(NEW,CATLG),UNIT=disk,
//          VOL=SER=nnnnnn,
//          DCB=(RECFM=VB,LRECL=150,BLKSIZE=1500),
//          SPACE=(TRK,(10,10),RLSE)
//SYSPCH     DD DSN=sort1,DISP=(NEW,CATLG),UNIT=disk,
//          SPACE=(TRK,1),DCB=BLKSIZE=80,VOL=SER=TECH01
```

idms.dba.loadlib	Dataset name of the CA-IDMS/DB library containing the DMCL and the database name table load modules
idms.loadlib	Dataset name of the CA-IDMS/DB library containing CA-IDMS executable modules
unload	DDname of the file containing data unloaded from the Total database (<i>total.data</i>)
total.data	Dataset name of the file containing data unloaded from the Total database
etotdmcl	SYSIDMS parameter specifying the name of the DMCL to be accessed at runtime ►► See the <i>CA-IDMS Database Administration</i> , for more information about SYSIDMS parameters.

etotal	SYSIDMS parameter specifying the name of the database to be accessed at runtime ►► See the <i>CA-IDMS Database Administration</i> , for more information about SYSIDMS parameters.
dcmsg	DDname of the CA-IDMS system message area (DDLDCMSG)
idms.sysmsg.ddldcmsg	Dataset name of the CA-IDMS system message area (DDLDCMSG)
reldctl	Dataset name for the file containing a control record with information about the subschema, DMCL, and segment to be used with FASTLOAD
disk	Symbolic device name
nnnnnn	Volume serial number of the file
sortdbl1	Output file from ETOTLOAD to be used as input for FASTLOAD
sort1	Punch file for DBL1 step

F.5 Assembling the CICS transparency interface (ETOTCINT)

Note: Any modifications to CA-IDMS load libraries should be applied by SMP/E (OS/390). Refer to *CA-IDMS Installation and Maintenance Guide - OS/390* for information on how to assemble and link edit a module using SMP/E. See Chapter 4, “Runtime Operations Phase” on page 4-1 for ETOTCINT syntax.

ETOTCINT (OS/390)

```
//          EXEC ASMFCL
//ASM.SYSLIB DD DSN=idms.srclib,DISP=SHR
//          DD DSN=cics.maclib,DISP=SHR
//          DD DSN=idms.maclib,DISP=SHR
//ASM.SYSIN DD *
ETOTCINT macro statement
END
//LKED.SYSLMOD DD DSN=idms.loadlib(etotcint),DISP=SHR
```

idms.srclib	Dataset name of the CA-IDMS/DB source library
cics.maclib	Dataset name of the CICS macro library
idms.loadlib	Dataset name of the CA-IDMS/DB load library
etotcint	Name of the ETOTCINT module

F.6 Creating the CICS CA-IDMS/DB Interface (IDMSINTC)

Initial Installation: When installing the CA-IDMS/TOTAL Transparency for the CICS environment, a CICSOPTS module will be assembled and link edited as part of module IDMSINTC. All parameters for CICSOPTS that are required for the TOTAL Transparency will be automatically generated by the CAISAG installation utility when you indicate the product is to be installed, either as part of an integrated solution or as a single product during an ADDON install. The IDMSINTC load module will include all modules specifically required to run the TOTAL Transparency.

Modifying the CICSOPTS: If you need to reassemble CICSOPTS to change any installation options, you must edit the CICSOPTS member in your SMP/E PPOPTION file and apply a usermod causing the module to be reassembled. A sample usermod, UMCOPTS, can be found in the SAMPJCL library installed with your library tape.

Important: Be sure PLT entries are created to execute IDMSINTC at CICS startup and shutdown. Additionally, be sure that PPT entries for CA-IDMS modules have been added to your CICS system. These are used by IDMSINTC in setting up the CA-IDMS environment.:

►► For more information on adding PPT entries, see the CA-IDMS Installation Guide for your operating environment. For information on the CICSOPT macro and its parameters, refer to *CA-IDMS System Operations*.

F.7 Batch application program link edit

Use this JCL to link edit the compiled Total application program with the ETOTBINT transparency module and the CA-IDMS/DB modules IDMSOPTI and IDMS.

BATCH APPLICATION PROGRAM LINK EDIT (OS/390)

```
//LKEDTOTA EXEC PGM=IEWL,PARM=(XREF,LET,LIST),REGION=256K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSLMOD DD DSN=idms.loadlib,DISP=SHR
//SYSLIB DD DSN=idms.loadlib,DISP=SHR
//userlib DD DSN=user.loadlib,DISP=SHR
//SYSLIN DD *
INCLUDE userlib(userprog)
INCLUDE SYSLIB(ETOTBINT)
INCLUDE SYSLIB(IDMS)
INCLUDE SYSLIB(IDMSOPTI) Optional
ENTRY userprog
NAME userprog(R)
/*
```

idms.loadlib	Dataset name of the CA-IDMS/DB load library
disk	Symbolic device name of the linkage editor work file
user.loadlib	Dataset name of the user load library containing the batch application program
userlib	DDname of the user load library containing the batch application program
userprog	Name of the Total batch application program

F.8 CICS application program link edit

Use this JCL to link edit the compiled Total application program with the ETOTCINT macro.

CICS APPLICATION PROGRAM LINK EDIT (OS/390)

```
//LKEDTOTA EXEC PGM=IEWL,PARM=(XREF,LET,LIST),REGION=256K
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD UNIT=disk,SPACE=(CYL,(2,2))
//SYSLMOD DD DSN=idms.loadlib,DISP=SHR
//SYSLIB DD DSN=idms.loadlib,DISP=SHR
// DD DSN=cics.loadlib,DISP=SHR
//userlib DD DSN=user.loadlib,DISP=SHR
//SYSLIN DD *
INCLUDE userlib(userprog)
INCLUDE SYSLIB(etotcint)
additional INCLUDE statements for CICS interface modules as required:
ENTRY userprog
NAME userprog(R)
```

idms.loadlib	Dataset name of the CA-IDMS/DB load library
cics.loadlib	Dataset name of the Total CICS load library
disk	Symbolic device name of the linkage editor work file
etotcint	Name of the ETOTCINT module
user.loadlib	Dataset name of the user load library containing the Total CICS application program
userlib	DDname of the user load library containing the CICS application program
userprog	Name of the CICS application program

F.9 Runtime JCL

Use the following JCL to execute a Total application program under the central version.

Runtime (OS/390)

```
//userprog EXEC PGM=userprog
//STEPLIB DD DSN=idms.dba.loadlib,DISP=SHR
//          DD DSN=idms.loadlib,DISP=SHR
//sysctl DD DSN=idms.sysctl,DISP=SHR
program input, as required
```

idms.dba.loadlib	Dataset name of the CA-IDMS/DB load library containing the DMCL and database name table load modules
idms.loadlib	Dataset name of the CA-IDMS/DB load library containing CA-IDMS executable modules
idms.sysctl	Dataset name of the SYSCTL file
sysctl	DDname of the SYSCTL file
userprog	Name of the Total application program

To execute Total application programs in local mode, remove the SYSCTL DD statement and insert the following statements after the STEPLIB DD statement:

```
//userdb DD DSN=user.userdb,DISP=SHR
additional database file assignments, as required
//sysjrn DD DSN=idms.tapejrn,DISP=(NEW,KEEP),UNIT=tape
```

userdb	DDname of the CA-IDMS/DB database file
user.userdb	Dataset name of the CA-IDMS/DB database file
sysjrn	DDname of the tape journal file
idms.tapejrn	Dataset name of the tape journal file
tape	Symbolic device name of the tape journal file

Appendix G. VSE/ESA Job Control Language

G.1 Generate (ETOTMAIN)

The following is the JCL you use to run the transparency generator (ETOTMAIN).

ETOTMAIN (VSE/ESA)

```
// LIBDEF *,SEARCH=idms.sublib
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
*
// DLBL  IDMSpch,'user.etotschm',1,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// DLBL  SYSPC03,'user.etotsubs',1,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// DLBL  SYSPC04,'user.etottbl',1,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// DLBL  SYSPC05,'user.etotload',1,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
*
// DLBL  SYSIDMS,'#SYSIPT'
// EXEC  ETOTMAIN,SIZE=448K
DMCL=etotdmc1
  FILENAME=SYSPC03 BLKSIZE=80
  FILENAME=SYSPC04 BLKSIZE=80
  FILENAME=SYSPC05 BLKSIZE=80
Input other SYSIDMS parameters, as required
/*
Total DDL source statements
User supplied control statements
ETOTTBL  source statements
ETOTLOAD source statements
/*
```

idms.sublib	The name of the CA-IDMS library
dcmsg	Filename of the system message area (<i>ddldcmsg</i>)
idms.sysmsg.ddldcmsg	File ID of the CA-IDMS system message area (DDLDCMSG)
SYSnnn	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial number
user.etotschm	Dataset name of the file containing the schema source generated by ETOTMAIN. Run this output through the schema compiler (IDMSCHEM).
user.etotsubs	Name of the file containing the subschema source generated by ETOTMAIN. Run this output through the subschema compiler (IDMSUBSC).
ssss	Starting track (CKD) or block (FBA) of the disk extent

1111	Number of tracks (CKD) or blocks (FBA) in the disk extent
nnnnnn	Volume serial number
user.etottbl	Name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL. Run this output through the assembly and link edit process.
user.etotload	Name of the file containing the customized loader generated by ETOTMAIN from the skeleton loader, ETOTLOAD. Run this output through the COBOL compile and link edit process.
etotdmcl	<p>SYSIDMS parameter specifying the name of the DMCL to be accessed at runtime</p> <p>►► See the <i>CA-IDMS Database Administration</i>, for more information about SYSIDMS parameters.</p>
SYSIDMS parameters	<p>Parameters you specify to establish a runtime environment.</p> <p>►► See the <i>CA-IDMS Database Administration</i>, for more information about SYSIDMS parameters.</p>

G.2 Assembling and link editing ETOTTBL

Use the following JCL to assemble and link edit ETOTTBL.

ETOTTBL (VSE/ESA)

```
// LIBDEF *,CATALOG=user.sublib
// DLBL IJSYSIN,'user.etottbl',1,SD
// EXTENT SYSIPT,nnnnnn
//      ASSGN SYSIPT,DISK,VOL=nnnnnn,SHR
// OPTION CATAL,NODECK,NOSYM
// PHASE etottbl,*
// EXEC ASMA90
/*
// EXEC LNKEDT
/*
CLOSE SYSIPT,SYSRDR
```

<code>user.sublib</code>	The name of the user sublibrary
<code>user.etottbl</code>	Name of the file containing the customized runtime control table generated by ETOTMAIN from the skeleton table, ETOTTBL. Run this output through the assembly and link edit process.
<code>nnnnnn</code>	Volume serial number
<code>etottbl</code>	Name of the customized transparency ETOTTBL load module, which replaces the Total DBMOD module

G.3 Compiling and link editing ETOTLOAD

Use the following JCL to compile and link edit ETOTLOAD.

ETOTLOAD (VSE/ESA)

```
// LIBDEF *,SEARCH=idms.sublib,CATALOG=user.sublib
// DLBL IJSYSIN,'user.etotload',1,SD
// EXTENT SYSIPT,nnnnnn
//   ASSGN SYSIPT,DISK,VOL=nnnnnn,SHR
// OPTION CATAL,NODECK,LIST,NOSYM,NOLISTX
// PHASE etotload,*
// EXEC FCOBOL
/*
// INCLUDE IDMSDBLU
// INCLUDE IDMS
// ENTRY etotload
/*
// EXEC LNKEDT
/*
CLOSE SYSIPT,SYSRDR
```

idms.sublib	The name of the CA-IDMS library
user.sublib	The name of the user sublibrary
user.etotload	Name of the the file containing the customized loader generated by ETOTMAIN
nnnnnn	Volume serial number
etotload	Name of the customized load program
	►► See the <i>CA-IDMS Database Administration</i> , for more information about SYSIDMS parameters.

G.4 Loading the database with ETOTLOAD

Use the following JCL to execute ETOTLOAD. Output from this step is processed by the FASTLOAD utility.

►►Refer to *CA-IDMS Utilities* for more information about the FASTLOAD utility statement.

ETOTLOAD (VSE/ESA)

```
*
// LIBDEF *,SEARCH=(user.sublib,idms.sublib)
// DLBL   dcmsg,'idms.sysmsg.ddldcmsg',1999/365,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
*
// DLBL  IDMSpch,'sort1',0,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1
// DLBL  SYS002,'sortdb11',0,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// DLBL  RELDCTL,'reldctl',0,SD
// EXTENT SYSnnn,nnnnnn,,ssss,5
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
*
// DLBL  unload,'total.data',0,SD
// EXTENT SYSnnn,nnnnnn,,ssss,1111
// ASSGN  SYSnnn,DISK,VOL=nnnnnn,SHR
*
// DLBL  SYSIDMS,'#SYSIPT'
// EXEC  ETOTLOAD,SIZE=256K
DMCL=etotdmc1 DBNAME=etotal
/*
/*
```

idms.sublib	The name of the CA-IDMS library
user.sublib	The name of the user sublibrary
dcmsg	Filename of the system message area (<i>ddldcmsg</i>)
idms.sysmsg.ddldcmsg	File ID of the CA-IDMS system message area (DDLDCMSG)
SYSnnn	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial number of the file
sort1	Punch file for DBL1 step
sortdb11	Output file from ETOTLOAD to be used as input for FASTLOAD
reldctl	File ID for the file containing a control record with information about the subschema, DMCL, and segment to be used with FASTLOAD
nnnnnn	Volume serial number

ssss	Starting track (CKD) or block (FBA) of the disk extent
1111	Number of tracks (CKD) or blocks (FBA) in the disk extent
unload	Filename of the file containing data unloaded from the Total database (<i>total.data</i>)
total.data	File ID of the file containing data unloaded from the Total database
etotdmcl	<p>SYSIDMS parameter specifying the name of the DMCL to be accessed at runtime</p> <p>►► See the <i>CA-IDMS Database Administration</i>, for more information about SYSIDMS parameters.</p>
etotal	<p>SYSIDMS parameter specifying the name of the database to be accessed at runtime</p> <p>►► See the <i>CA-IDMS Database Administration</i>, for more information about SYSIDMS parameters.</p>

G.5 Assembling the CICS/transparency interface (ETOTCINT)

The following is the JCL you use to assemble the CICS/transparency interface (ETOTCINT).

►► See Chapter 4, “Runtime Operations Phase” on page 4-1 for ETOTCINT syntax.

ETOTCINT (VSE/ESA)

```
*
// LIBDEF *,SEARCH=(idms.sublib)
// DLBL    IJSYSPH,'user.txtfile',0
// EXTENT  SYSPCH,nnnnnn,,ssss,1111
//        ASSGN  SYSPCH,DISK,PERM,VOL=nnnnnn,SHR
// OPTION  DECK,NOLINK,NOEDEC,LIST,NORL,NOXREF
// EXEC    ASSEMBLY
//        PUNCH  'ACCESS SUBLIB=idms.sublib'
//        PUNCH  'CATALOG etotcint.OBJ REPLACE=YES'
ETOTCINT macro statements
        END
/*
//        CLOSE  SYSPCH,00D
// DLBL    IJSYSIN,'user.txtfile',0
// EXTENT  SYSIPT,nnnnnn
//        ASSGN  SYSIPT,DISK,PERM,VOL=nnnnnn,SHR
// EXEC    LIBR,PARM='MSHP'
/*
//        CLOSE  SYSIPT,SYSRDR
/*
```

idms.sublib	The name of the CA-IDMS library
nnnnnn	Volume serial number
ssss	Starting track (CKD) or block (FBA) of the disk extent
1111	Number of tracks (CKD) or blocks (FBA) in the disk extent
etotcint	Name of the ETOTCINT module
user.txtfile	Intermediate work file output from the assembly step

G.6 Creating the CICS CA-IDMS/DB interface (IDMSINTC)

Initial Installation: When installing the CA-IDMS/TOTAL Transparency for the CICS environment, a CICSOPTS module will be assembled and link edited as part of module IDMSINTC. All parameters for CICSOPTS that are required for the TOTAL Transparency will be automatically generated by the CAIIMP installation utility when you indicate the product is to be installed, either as part of an integrated base install or as a single product during ADDON install. The IDMSINTC phase will include all of the modules specifically required to run the TOTAL Transparency.

Modifying CICSOPTS: If you need to reassemble CICSOPTS to change any installation options, you must edit the CICSOPTS member, reassemble your CICSOPTS module and relink IDMSINTC. Job control to do this should be taken from the job control that was generated by CAIIMP for your initial base tape installation.

Important: Be sure PLT entries are created to execute IDMSINTC at CICS startup and shutdown. Additionally, be sure that PPT entries for CA-IDMS modules have been added to your CICS system. These are used by IDMSINTC in setting up the CA-IDMS environment.:

►► For more information adding PPT entries, see the CA-IDMS Installation Guide for your operating environment. For information on the CICSOPTS macro and its parameters, refer to *CA-IDMS System Operations*.

G.7 Batch application program link edit

Use the JCL below to link edit a batch application program.

Batch Application Program Link Edit (VSE/ESA)

```
// LIBDEF *,SEARCH=(idms.sublib),CATALOG=user.sublib
// OPTION CATAL
  PHASE userprog,*
  INCLUDE userprog
  INCLUDE ETOTBINT
  INCLUDE IDMS
  INCLUDE IDMSOPTI                               Optional
  ENTRY userprog
// EXEC LNKEDT
/*
```

<u>idms.sublib</u>	The name of the CA-IDMS library
<u>user.sublib</u>	The name of the user sublibrary
<u>userprog</u>	Name of the Total batch application program

G.8 CICS application program link edit

Use the following JCL to link edit a CICS application program.

CICS Application Program Link Edit (VSE/ESA)

```
// LIBDEF *,SEARCH=(idms.sublib),CATALOG=user.sublib
// OPTION CATAL
  PHASE userprog,*
  INCLUDE userprog
  INCLUDE etotcint
Additional INCLUDES for CICS interface modules, as required
  ENTRY userprog
// EXEC LNKEDT
/*
```

idms.sublib	The name of the CA-IDMS library
user.sublib	The name of the user sublibrary
userprog	Name of the Total CICS application program
etotcint	Name of the ETOTCINT module

G.9 Runtime JCL

Use the following JCL to execute a Total application program under the central version.

Runtime (VSE/ESA)

```
// LIBDEF *,SEARCH=(user.sublib,idms.sublib)
// DLBL sysctl,'idms.sysctl',1999/365,SD
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL SYSIDMS,'#SYSIPT'
// EXEC userprog
Optional SYSIDMS parameters
/*
Program input if any
/*
```

user.sublib	The name of the user sublibrary
idms.sublib	The name of the CA-IDMS library
sysctl	Filename of the SYSCTL file
idms.sysctl	File ID of the SYSCTL file
SYSnnn	Logical unit of the volume for which the extent is effective
nnnnnn	Volume serial number
userprog	Name of the Total application program
SYSIDMS parameters	Parameters you specify to establish a runtime environment.
	►► See the <i>CA-IDMS Database Administration</i> , for more information about SYSIDMS parameters.

Executing in local mode: To execute user application programs in local mode, remove the SYSCTL statements, and insert the following statements before the EXEC *userprog* statement:

```
// DLBL dcmmsg,'idms.sysmsg.ddldcmmsg',1999/365,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
// DLBL userdb,'user.userdb',,DA
// EXTENT SYSnnn,nnnnnn
// ASSGN SYSnnn,DISK,VOL=nnnnnn,SHR
Additional database file assignments, as required
// TLBL sysjrn1,'idms.tapejrn1',,nnnnnn,f
// ASSGN sys009,TAPE,VOL=nnnnnn
```

dcmmsg	Filename of the system message area (<i>ddldcmmsg</i>)
--------	--

idms.sysmsg.ddldcmsg	File name of the CA-IDMS system message area (DDLDCMSG)
userdb	Filename of the user CA-IDMS/DB database file
user.userdb	File ID of the user CA-IDMS/DB database file
SYSnnn	Logical unit assignment of the user CA-IDMS/DB database file
sysjrn1	Filename of the tape journal file
sys009	Logical unit assignment of the tape journal file
idms.tapejrn1	File ID of the tape journal file
nnnnnn	Volume serial number
f	File number of the tape journal file

Index

A

ADDV 1-18
application program
 batch - OS/390 JCL F-12
 batch - VSE/ESA JCL G-10
 CICS - OS/390 JCL F-13
 CICS - VSE/ESA JCL G-11
 modification 4-3
area 2-9
AREA statement 2-8—2-9
 example 2-8
 syntax 2-8

C

CALC key 1-13
CICS
 error code D-8
 ETOTCINT 1-7
 IDMSINTC 1-7
 user exit module A-8
CICS CA-IDMS/DB interface
 OS/390 JCL F-11
CICS transparency interface
 OS/390 JCL F-10
CICS/transparency interface
 VSE/ESA JCL G-8
CLOSX 1-16
CNTRL 1-15
COMIT 1-15
comments 2-3
compile 3-3
components 1-5
components of the transparency 1-5
Creating IDMSINTC
 CICS G-9
CSITULOD 2-15, 3-19

D

data definition
 generate 3-3
data description 1-9
 AREA statement 2-8—2-9
 general discussion 2-3—2-4
 LOAD statement 2-15—2-17
 SCHEMA statement 2-9—2-11
 steps 2-3

data description (*continued*)
 SUBSCHEMA statement 2-11—2-12
 SYNONYMS statement 2-6—2-8
 USAGE-MODE statement 2-12—2-15
data item 1-13
data structures 1-13—1-14
 figure 1-5
database
 administration functions 1-15
 initialization 3-3, 3-18
 load 3-3, 3-20
 loader - OS/390 JCL F-6, F-8
 loader - VSE/ESA JCL G-5
 selection 2-3
 structure 1-14
 unload 3-19
database conversion 3-3
 figure 1-11
 generator messages B-3—B-8
 large database E-3
 loader messages C-3
database conversion phase 1-9
database conversion programs 1-9
database name table 2-4
 compile 3-3
 generate 3-3
 link 3-3
 punch 3-3
DATBASAF A-3
 linkage conventions A-3
DATBASBF A-3
 linkage conventions A-3
 sample A-4, A-8
DDL
 physical 2-4
DDL statement
 limits 2-3
 preparation 2-3
DMCL 2-4
 compile 3-3
 generate 3-3
 link 3-3
 preparation 3-14
 punch 3-3
DMCL compiler
 output 3-15

E

- element 1-13
- ENDLG 1-15
- ENDTO 1-17
- environments supported 1-4
- environments supported VSE/ESA operating environments
 - and supports 1-4
- error messages
 - See* error-status code cross reference
 - See* generator
 - See* loader
 - See* runtime error codes
- error-status code cross reference D-5—D-7
- ETOT07EX A-3
- ETOT56EX A-3
- ETOTBINT 1-16, 4-4
- ETOTCINT 1-7, 4-3, 4-4, G-8
 - OS/390 JCL F-10
 - syntax 4-3
 - VSE/ESA JCL G-8
- ETOTLOAD 1-7, F-6, F-8, G-5
 - See also* transparency loader
- ETOTMAIN 1-5, G-1
 - See also* transparency generator
- ETOTROO2 E-3
- ETOTROOT E-3
- ETOTTBL 1-7
 - See also* runtime control table
 - VSE/ESA JCL G-4
- ETOTTRAN 1-7, 4-4

F

- features
 - supported 1-15, 1-16
 - unsupported 1-16, 1-18
- file 1-13, 2-7, 2-9
 - master 1-14
 - variable 1-14
- FINDX 1-17
 - SKIP option 1-17
- FORMAT utility 3-3
- FREEX 1-15
- functions
 - unsupported 4-3

G

- generator 1-5, 3-3, 3-4
 - control statement limits 2-6
 - control statement syntax 2-6

generator (*continued*)

- control statements 2-4
- error messages B-6
- fatal messages B-7—B-8
- input 1-5, 3-4
- input limits E-3
- output 3-4
- SCHEMA statement 2-9
- SUBSCHEMA statement 2-11
- warning messages B-7

I

- IDMSBCF
 - input 3-15
 - output 3-15
- IDMSCALC 1-16
- IDMSCHEM
 - See* schema compiler
- IDMSDMCL
 - See* DMCL compiler
- IDMSINTC 1-7, 4-4, F-11
- IDMSOPTI 4-4
- IDMSUBSC
 - See* subschema compiler
- implementation 1-9—1-18

J

- JCL
 - ETOTMAIN G-1
 - OS/390 F-3—F-14
 - VSE/ESA G-1—G-13
- journaling 1-15

L

- limits 2-3, 2-6
- linkpath 1-13
 - conversion to set 1-13
 - field 1-13, 1-14, 2-7, 2-15
 - primary 1-14
- LOAD statement 2-15—2-17
 - example 2-17
 - syntax 2-15
- loader 1-7, 2-3, 2-4, 3-3, 3-20
 - generate 3-3
 - messages C-3
- logging 1-15, 1-17
- logical database description 2-4

M

MARKL 1-17
master file control key 1-13, 2-7, 2-15
member 1-14
messages
 error-status code cross reference D-5—D-7
 generator B-3—B-8
 loader C-3

N

name
 size limits 2-6

O

OPENX 1-16
OS/390 JCL F-3—F-14
owner 1-14

P

physical database
 preparation 3-14
 store 3-3
physical DDL 2-4
pilot project 2-3
pointer 1-13, 1-14

Q

QMARK 1-15
QUIET 1-15

R

RDNXT 1-17
 SKIP option 1-17
record
 coded variable 1-18
 prefix 1-14
 type 1-13
recovery 1-15, 1-17
relationships 1-14
releases (Total)
 supported 1-15, A-3
RQLOC 1-16, 1-17
RSTAT 1-17
runtime
 OS/390 JCL F-14
 VSE/ESA JCL G-12

runtime control table 1-7, 2-3, 2-4, 4-3
 OS/390 JCL F-5
 VSE/ESA JCL G-4
runtime error codes
 detected by CA-IDMS/DB D-5—D-7
 internal transparency D-8—D-9
 Total syntax errors D-4
 US D-9
runtime interface
 area usage mode 2-12
 ETOTCINT 1-7
 ETOTTRAN 1-7
 general discussion 1-7
 IDMSINTC 1-7
runtime operations 4-3—4-4
 error-status code cross reference D-5—D-7
 figure 1-11
 steps 4-3
 US status codes D-9
runtime operations phase 1-11

S

schema 2-9
schema compiler 3-13
 input 3-13
 output 3-13
SCHEMA statement 2-9—2-11
 example 2-11
 syntax 2-9
set 1-13, 1-14
 membership 1-14
 order 1-14
 order with ETOTLOAD 3-20
 structure 1-3
SINOF 1-16
SINON 1-16
 REST parameter 1-17
subschema
 compile 3-3
 link 3-3
 load module 3-3, 3-16
 preparation 3-3, 3-16
subschema compiler 3-16
 input 3-16
 output 3-16
SUBSCHEMA statement 2-11—2-12
 example 2-12
 syntax 2-11
SYNONYMS statement 2-6—2-8
 example 2-8

SYNONYMS statement (*continued*)

 syntax 2-7

T

Total data structures

See data structures

transparency generator

 OS/390 JCL F-3

 VSE/ESA JCL G-2

transparency loader

 compile and link edit G-5

 OS/390 JCL F-6, F-8

 VSE/ESA JCL G-5

U

unload 3-3

US status codes D-8, D-9

USAGE-MODE statement 2-12—2-15

 examples 2-14

 syntax 2-12

user exits A-3

 batch exit module A-4

 CICS exit module A-8

 DATBASAF A-3

 DATBASBF A-3

V

VIA 1-13, 1-14

VSE/ESA JCL G-1—G-13

W

WRITD 1-17

